# THE ELECTRONIC WARFARE APPLICATION OF SPECIAL PURPOSE MICROPROGRAMMED MINICOMPUTERS

Paul David Frazer

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California


# THESIS

THE ELECTRONIC WARFARE APPLICATION
OF
SPECIAL PURPOSE MICROPROGRAMMED MINICOMPUTERS

by

Paul David Frazer

June 1974

Thesis Advisor: S. Jauregui, Jr.

Approved for public release; distribution unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>The Electronic Warfare Application of Special Purpose Microprogrammed Minicomputers | | 5. TYPE OF REPORT & PERIOD COVERED<br>Master's Thesis; June 1974<br>6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Paul David Frazer | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Naval Postgraduate School<br>Monterey, California 93940 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Naval Postgraduate School<br>Monterey, California 93940 | | 12. REPORT DATE<br>June 1974<br>13. NUMBER OF PAGES<br>114 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>Naval Postgraduate School<br>Monterey, California 93940 | | 15. SECURITY CLASS. (of this report)<br>Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An algorithm for the application of a special purpose minicomputer to the automatic analysis and initial classification of EW signals is developed. The minicomputer assembly language version of this algorithm is presented. A FORTRAN program which generates realistic data for signal analysis is described. Using the simulated data and an instruction level simulation program,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

the algorithm is tested using an IBM 360 computer. Micro-programming techniques are presented which optimize the minicomputer application.

# The Electronic Warfare Application of Special Purpose Microprogrammed Minicomputers

by

Paul David Frazer
Lieutenant Commander, United States Navy
B.E.E., Marquette University, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1974

# ABSTRACT

An algorithm for the application of a special purpose minicomputer to the automatic analysis and initial classification of EW signals is developed. The minicomputer assembly language version of this algorithm is presented. A FORTRAN program which generates realistic data for signal analysis is described. Using the simulated data and an instruction level simulation program, the algorithm is tested using an IBM 360 computer. Microprogramming techniques are presented which optimize the minicomputer application.

TABLE OF CONTENTS

# I. STATEMENT OF THE PROBLEM

Electronic Support Measures (ESM) are designed to extract information about hostile forces from the electromagnetic environment they create. Since the earliest days of radar, there has been a parallel development of equipment, techniques and tactics designed to counter the use of radar by hostile forces by using the transmitted energy to detect, locate and track the hostile force, often without your own unit being detected by the searching radar system.

The current proliferation of radar systems, telemetry systems, communications systems and navigation systems which all use electromagnetic radiation for operation has created a condition of overcrowding of the electromagnetic spectrum. More important from the ESM standpoint is that the detection of a signal at almost any reasonable frequency is assured. What is not assured is being able to detect signals which are determined to be of interest in a sufficiently timely manner to be of use to the Operational Commander.

The dense signal environment which exists in world areas of interest, coupled with the rapidly decreasing allowable reaction times imposed by new and developing weapons systems, and the continuous requirement to maintain viable capabilities with a shrinking manpower force has imposed a requirement for faster, more accurate, automatic classification of ESM signals. The development of techniques for

their employment in the task of signal analysis in the ESM environment.

To date, the major effort has been to apply digital computers to the task of off-line signal analysis and specific threat identification problems. What is now required is the expansion of these or similar techniques to handle the task of routine surveillance over a wide frequency range, filtering received information for interest, determination of the parameters of received signals, initial classification and threat evaluation, and display to an operator or output to a remote location. All of this has to occur in real time with a high probability of developing correct information. It is felt that such a system may be feasible if it is designed taking advantage of the rapid data handling capability of digital computer systems specifically designed for this application.

## II. CONCEPTUAL SYSTEM DESIGN

### A. SYSTEM OPERATIONAL REQUIREMENTS

A system designed to fulfill the needs of automatic ESM surveillance classification and threat analysis has the following characteristics for effective performance.

a. Broad frequency range.

b. Simultaneous reception of all frequencies within the frequency range.

c. Means of determining received signal characteristics within a reasonable accuracy.

d. Ability to operate successfully in dense environments with tens or hundreds of thousands of pulses per second being received.

e. Ability to associate received pulses and other forms of radiation into pulse trains or other forms for identification and analysis.

f. Means of determining the bearing of the received energy to within acceptable accuracy.

g. Accurate means of identifying signals as to class and probable source.

h. High probability of intercept.

i. Low false alarm rate.

### B. SYSTEM COMPONENTS

Conceptually such a system could be assembled using a series of Instantaneous Frequency Measurement (IFM) receivers

that would cover the frequency bands of interest. The outputs of these receivers would be applied to small special purpose digital computers which would perform the initial information filtering task of creating a data set of the parameters of all of the received pulse trains and exclude from further processing those signals which are obviously of no interest to the tactical environment. Information thus derived is then to be passed to a second computer system at considerably lower data rates which will allow detailed analysis of signals, final classification tasks and automatic threat evaluation.

Detailed design of an entire system such as that described is beyond the scope of this thesis. The problem area specifically treated is the first interface between the IFM and the associated digital computer and the information filtering task it is required to do.

## C. SPECIFIC SUBSYSTEM DESIGN ANALYSIS

The specific design problem analyzed herein is stated as follows.

Given an IFM receiver with an omnidirectional antenna and reasonable sensitivity and probability of detection at all frequencies within its band width, analyze the feasibility of providing a special purpose digital computer that will accept all information developed by the IFM and provide as real-time outputs the frequency, pulse width, pulse repetition interval, scan type and the time of last intercept.

## III. COMPUTER SELECTION FOR PURPOSES OF FEASIBILITY ANALYSIS

To complete the analysis of the feasibility of the automatic IFM system, it was desirable to actually encode and simulate the operation of the system using an existing computer instruction set. It was also of interest to determine the required execution times, pinpoint improvements to the basic algorithm, establish required additional 'special case' instructions which could improve the efficiency of execution, and establish desirable hardware arrangements which would enhance the overall operation of the computer. Through the experience of operating a simulation program and the test of various software and hardware techniques, the detailed specifications for a special purpose computer can be determined. This computer would be specifically designed and optimized for the demanding high data rates anticipated. Investigation disclosed the existence of a small high speed special purpose computer produced by Applied Technology Corporation (ATI) in Sunnyvale, California. This computer is designated as the Applied Technology Airborne Computer (ATAC) and is a 16 bit fixed point 2's complement machine. A number of unique features of this computer made it an obvious choice for this study. The general computer characteristics and these special features are summarized below. Further detailed information is available in ATAC, Applied Technology Airborne Computer, Vol. I, Principles of Operation.

1. 16-bit instruction word length with a 437 nanosecond major cycle time.

2. Sixteen general registers.

3. Four basic instruction addressing modes with three supplemental modes allowing a wide range of direct, indexed and immediate execution instructions.

4. Asynchronous memory with provisions of up to 4 Direct Memory Access channels.

5. Memory types can be mixed in any combination of Read Only Memories (ROM), Bipolar Ramdom Access Memories (RAM), Read Mostly Memories (RMM) and slower core memory devices.

6. Twelve main priority interrupts.

7. 16-bit parallel input/output operations.

8. Extremely small size and power requirements: 6 pounds, 115 cubic inches, and a power consumption of approximately 70 watts for a nominal configuration.

9. An existing software support package designated as the ATAC Programming Support System (APSS).

10. Microprogrammed instruction set allowing ease of modification of the instruction set within the hardware limitations of the machine.

11. Computer assisted hardware design and fabrication allowing relative ease and speed of modification of the hardware configuration (within limits) to permit optimization of the computer for specific applications.

The features listed under 9, 10 and 11 are the essential features required for this feasibility study. As the basic

speed, memory configuration and programming flexibility required are provided by the existing ATAC, arrangements were made with ATI to allow use of portions of the APSS for this effort.

A. ATAC PROGRAMMING SUPPORT PACKAGE (APSS).

APSS is a computer simulation program designed and programmed for use on IBM 360/370 series Operating Systems. The complete package consists of the following items.

Volume 1 - ATAC BASIC PRINCIPLES OF OPERATION

This document describes the basic ATAC machine characteristics and the instruction set.

Volume 2 - ATAC PROGRAMMING SUPPORT SYSTEM (APSS)

APSS is the support software system for ATAC which allows preparation and debugging of programs via an IBM 360/370 computer system. The documents in this volume describe how to set up and use the APSS system and includes publications describing the PL/ATAC Compiler, Assembler, Loader Simulator and microprogramming facilities.

Section I - APSS Job Management

Section II - PL/ATAC Compiler

Section III - APSS Assembler/Loader

Section IV - APSS Instruction-Level Simulator

Section V - APSS Microprogramming Software

Volume 3 - ATAC SYSTEM DIAGNOSTIC PROGRAMS

This document summarizes the use of ATAC's Diagnostic programs for instruction test, I/O test, interrupt system checkout, and condition code test.

Volume 4 - ATAC CIRCUIT BOARD TEST PROGRAMS

This document summarizes the ATAC circuit board test procedures.

Volume 5 - ATAC OPERATOR'S GUIDE

Procedures for the operation of an ATAC computer are described in this document.

Volume 6 - ATAC 16A MICROPROGRAMMING AND SIMULATION

Procedures for developing microprograms for the ATAC 16A central processing unit.

The portions of APSS utilized for this study are listed below. Other volumes were not required but their existence is noted. Future efforts in the development of specifications for a special purpose computer should be made with the knowledge that such systems exist. The advantages accrued by having a computing system specifically tailored to the task at hand are self evident. Additional flexibility is provided through the use of microprogramming in that a completely different instruction set can be defined by replacing the microprogram Read Only Memory allowing the same basic machine to be optimized for several different tasks and reconfigured by changing a small inexpensive module.

Volume 1 - ATAC BASIC PRINCIPLES OF OPERATION

Volume 2 - ATAC PROGRAMMING SUPPORT SYSTEM (APSS)

Section I APSS Job Management

Section III APSS Assembler/Loader

Section IV APSS Instruction Level Simulator

The APSS Instruction Level Simulation Program is currently a resident program on the IBM 360/67 at the Naval Postgraduate School. Details concerning the implementation of the program are contained in Appendix A. Details concerning its use are contained in Appendix B.

## IV. THE IFM DIGITAL PROCESSOR

The initial design concept for the interface between the IFM and the special purpose digital computer is a common memory storage area to which both units have Direct Memory Access on a non-interrupt basis. The data to be passed between the IFM and the computer consists of the frequency, pulse width and time of arrival of all received signals. Using this information, the computer can classify signals as pulsed type radar signals based on the indicated pulse width. Further analysis is conducted by establishing initial pulse files segregated by frequency and ordered by the sequence in which they were received. An estimate of the PRI is made by averaging the time between the arrival of adjacent pulses in the file. Small variations in the PRI estimate are treated as noise. Large variations in the time between pulses are considered caused by antenna scan characteristics. Variations between these extremes are considered due to the superposition of multiple pulse trains or due to a complex PRI 'jitter' imposed on the emitter. Variations due to noise and antenna scan are handled internally. Variations due to interleaved pulse trains and complex jitter patterns are signalled and further analysis of these cases is handled by the main classification computer. The design calls for the special purpose IFM computer to output the following information for each pulse train received.

a. Frequency

b. Pulse width within limits, the value

c. Pulse width out of limits, a designation

d. Stable PRI, the value

e. Unstable PRI, a designation

f. Antenna scan time

g. Time of last update

h. When ordered, a pulse train of specified length to be used for analysis of complex signals by the main classification computer.

## A. TIME ENCODING

Determination of the pulse width and the time of arrival requires a digital clocking system whose output is directly translatable into real time. For purposes of analysis it is assumed that when the IFM threshhold is exceeded in a specific frequency band, the current value of the clock is loaded into the time of arrival words. Simultaneously the pulse width word is cleared and then incremented at the clock rate. When the pulse energy ceases, the pulse width word contains the pulse width scaled by the clock rate.

The clock consists of a 40 bit counter. The clock rate is 10 Megahertz. This corresponds to a least significant bit value for pulse width of 0.1 microseconds. The pulse width word consists of bits 0-15 of the counter which allows representation of pulse widths of up to 6.5 milliseconds treating the most significant bit as a magnitude

instead of a sign bit. The time of arrival double precision word consists of bits 8-39. This corresponds to a least significant bit value of 25.6 microseconds. The maximum clock time (time to overflow) is 15.3 hours. The most significant bit of the clock is treated as a sign bit to allow detection of the data out of sequence condition.

Accuracy is determined by the stability of the basic clock oscillator. Measurement is made to within 0.05 microseconds for pulse width and to within 12.8 microseconds for PRI.

## B. THE COMMON MEMORY BLOCK

The common memory block is used as a large buffer to match the differing data rate handling capability of the IFM and the computer. For purposes of analysis, it is assumed that the IFM can respond and prepare a series of four data words in the times achievable using TTL logic circuitry. This allows the IFM to load the double precision words for time of arrival and the word for frequency first, the word for pulse width as soon as the pulse has completely been received and accomplish the entire transfer of the four words in approximately 3 microseconds or the pulse width of the received signal + 0.5 microseconds which ever is longer. For signals with pulse widths of less than 3 microseconds, the IFM data will have a maximum burst rate of approximately 300 KHZ.

Simulations conducted using the algorithm described in Chapter IV indicate the average time required for the computer

to process each received pulse including read and write times is approximately 45 microseconds with improvement to 35-40 microseconds considered readily possible. This corresponds to a maximum average data rate of approximately 22,000 pulses per second.

The IFM loads data words into the common memory block sequentially starting from the '0' location and wraps around when the block is full. Also starting from the '0' location, the computer accesses the data words and associates them with the appropriate pulse train and then accesses the next word. The size of the buffer establishes the length of the time that transitory high data rates can be sustained by the system before the computer is 'lapped' by the IFM input. In this case, the computer is always accessing the most recent data set and the information stored by the IFM previously is lost. When the data rate decreases, it may become possible for the computer to catch up to the IFM to remove the lap condition. In this case, the data read by the computer will be out of sequence and will be discarded.

Pulse information is stored in the common memory block in a series of four words per pulse. These words are defined as follows.

a. Frequency. Frequency is represented by an integer corresponding to the frequency bin in which the detection occurs.

b. Pulse width. The pulse width is represented by an integer which is a scaled time value.

c. Time of arrival. Time of arrival is expressed as a double precision integer, the lower half of which is the third word, the upper half the fourth word.

The computer can keep track of the existence of new data in the word block associated with an individual pulse by resetting the frequency word to 0 after it is read. By testing the frequency word, the computer establishes whether the IFM has loaded new data since its last look at this location.

By making the IFM output and the computer input completely independent and asynchronous in operation, timing and interrupt communications, with their high time overhead costs are avoided.

## C. THE IFM COMPUTER ALGORITHM

An algorithm has been developed which defines the operations required to develop pulse trains and extract PRI information from the IFM output data. This algorithm causes the computer to read a memory location and determine if data is available. If the data is available, the frequency word is used as a relative address to enter a file which provides linking information for the pulse train files developed for that frequency. This file data word is utilized as two half words. The lower half is used as a pulse train file pointer. The upper half is used as a pointer for an overflow storage area in the event there are two or more pulse trains associated with a given frequency.

The primary means of pulse train association is by frequency discrimination.

The pulse train file pointer is used as a relative address for entry into a contact file entry table. The table entry corresponding to the pulse train file pointer contains two words. The first word contains the frequency of the pulse train for cross reference purposes. The second word contains the absolute entry address of the actual file.

The pulse train file is used both to accumulate data for calculation of parameters and to provide output information to the main classification computer by using Direct Memory Access techniques. The file is a block of 64 memory locations which are treated as 16 four word groups. This allows storing information on 16 consecutive pulses which are used to develop an estimate of the PRI. Before 16 pulses have been received, the file is said to be building. After 16 pulses have been received and the PRI estimated, the file is said to be closed. If no current pulse information is contained in the file, it is said to be empty. The distinction is required because the contents of the words of the file have different meanings for the different cases. The contents of the file words are described below. The choice of 16 pulses used for PRI determination is arbitrary and additional testing and evaluation is required to determine the optimum number of pulses considering detection probabilities, normal antenna scan rates, false alarm rates, and simulation and operational experiences.

1. File building.

a. Word 1. Pulse width of the initial pulse detected when the pulse train file was established.

b. Word 2. Counter indicating the number of pulses stored in this file. The counter counts from -16 to 0.

c. Word 3, 4. Double precision time of arrival of the first pulse.

d. Word 5. Pulse width of the second pulse.

e. Word 6. Time difference between the arrival of the second pulse and the first.

f. Word 7, 8. Double precision time of arrival of the second pulse.

g. Succeeding groups of 4 words are arranged in a similar fashion with the second word of the group containing the time interval between the pulse and the one preceding it.

2. File closed.

a. Word 1. Average pulse width of the file.

b. Word 2. A positive integer represents the average PRI. If the location contains a zero, a stable PRI cannot be determined.

c. Word 3, 4. Double precision time of last information update.

d. Word 5, 6. Double precision time of arrival of the first pulse of the latest consecutive group of pulses to be received.

e. Word 6, 7. Double precision antenna scan rate.

3. File empty.

The file empty condition is indicated by the first word of the contact file entry file being zero. The contents of the pulse train file should be considered as being unspecified.

The scheme of using the frequency as a relative entry address to the frequency file, using the data contained in the frequency file as an entry into a contact file and using the data contained in the contact file to provide the actual address of the pulse train file is required to maintain pulse train files without the requirement to have a separate pulse train file for every possible combination of frequency and pulse width. A total of 64 pulse train files can be established and maintained simultaneously in the current configuration.

In the event that two pulse trains of differing pulse widths are received in the same frequency bin, provision has been made to discriminate these signals by pulse width and establish and maintain separate files on these signals. If interleaved pulse trains of nearly equal pulse widths are received on the same frequency, the algorithm will consider the pulse train as having a complex PRI and enter a zero in the PRI word. The main classification computer can

then designate the requirement to establish a file of sufficient length to collect enough pulses for analysis by inserting the required number of pulses into the PRI word. After filling 60 of the 64 assigned locations for the pulse train file, the remainder of the long file is inserted into an overflow section of memory. Words 61-64 are used to provide linking information.

Calculation of the PRI after 16 pulses have been received is accomplished by averaging the time interval between pulses. Sixteen pulses are contained in the pulse file and the seventeenth pulse to arrive causes the start of this calculation to provide 16 time intervals for the averaging. The intervals are treated in reverse order of their appearance. The latest interval is treated as a standard and the previous interval is compared with it. If the two intervals are not nearly equal, the condition of one being nearly equal to a multiple of the other is check for multiples up to 4. If this condition is met, both intervals are made equal to the smaller interval and their sum is accumulated. If the condition is not met, both intervals are still made equal to the smaller interval (the longer interval discarded) their sum accumulated and a discard counter is incremented. The process is repeated with the other time of arrival increments in the file until all 16 intervals have been summed or until the discard counter reaches 4. If 4 pulses are discarded, the PRI is considered unstable and a zero is inserted into the PRI word. If all 16 pulse intervals are

summed, the resulting sum is divided by 16 (shifted right 4 bits) and the result inserted into the PRI word. Concurrently, the pulse widths are also summed to calculate the average pulse width of the file in a similar manner.

After the file has been closed, succeeding pulses received are utilized to update the data in the file. If a received pulse width is within limits of the file pulse width, it is compared with the file pulse width to determine its relative size. If the new one is smaller, a 1 is subtracted from the file pulse width. If larger, a 1 is added. This feature ensures that long term drifts in the parameter value can be tracked. If a PRI has been established for the file, and the time interval between succeeding pulses exceeds established limits, the new pulse is considered to have established simple antenna scan parameters and an anetnna scan rate estimate is calculated.

The IFM computer algorithm is shown in flow chart form in Figure 1.

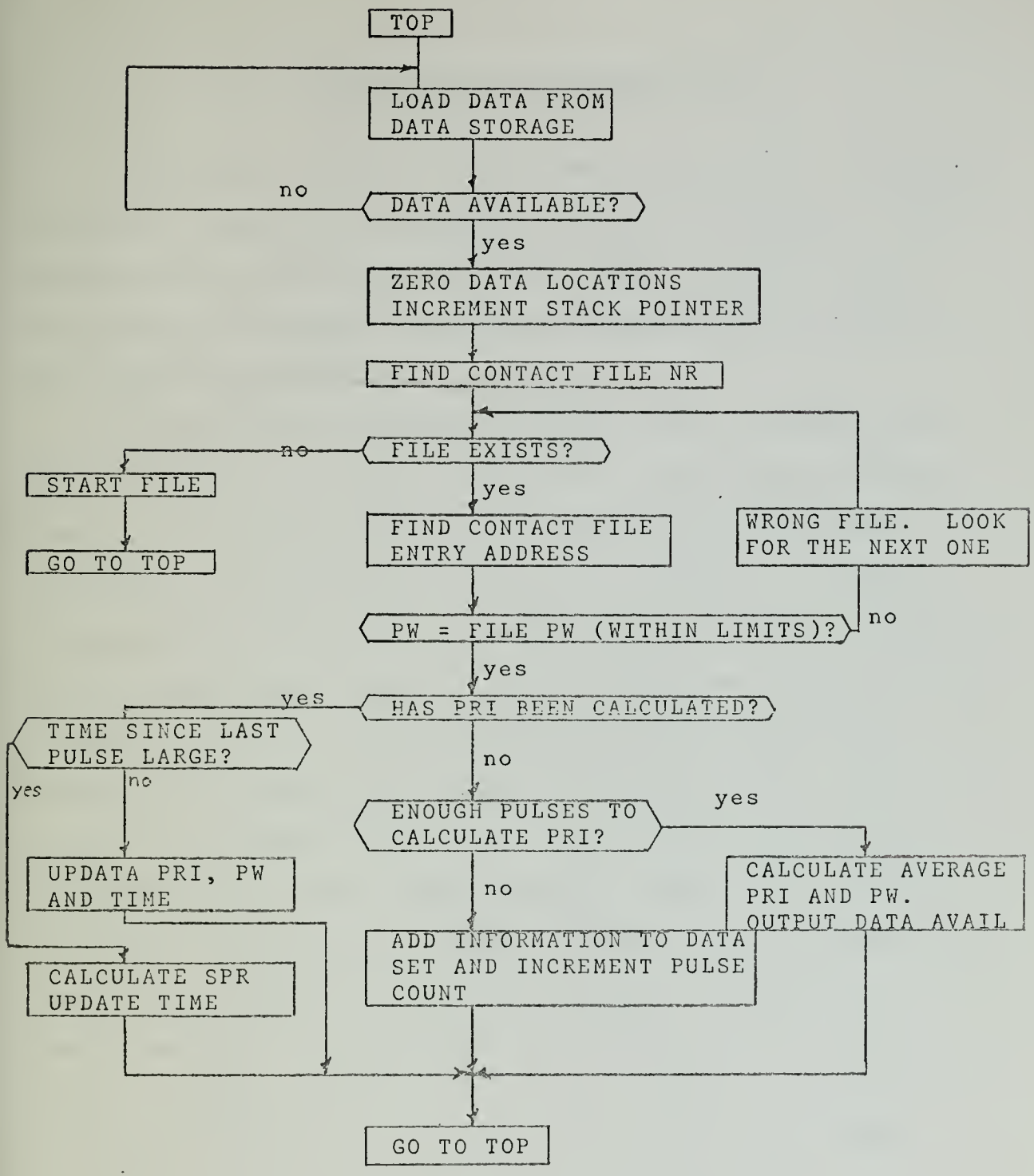



Figure 1. THE IFM DIGITAL PROCESSOR ALGORITHM (SIMPLIFIED FLOW CHART)

## V. DATA SIMULATION PROGRAM (DSP)

The DSP was written by the author to provide a relatively realistic input data environment to the ATAC Simulater. The program was written well before the ATAC Simulater became available and includes many features which are not useful in its current application.

The DSP provides simulated signal levels arriving at an IFM receiver in a dynamic environment. Input data provides initial set up positions for the IFM and up to 10 emitter platforms. The IFM and all of the 10 platforms are given courses and speeds which are used to update their positions as the simulation proceeds. Each platform may have up to 5 emitters on board of any specified frequency, PRI, PW, anetnna scan, antenna gain, and antenna beam width. The program establishes the initial antenna position by using a uniformly distributed random variable, updates the antenna position, calculates the signal strength of each transmitted pulse when received by the IFM and outputs those pulses which are above the specified threshold sensitivity of the IFM. The output consists of the time sequence of pulses received by the IFM including the frequency, pulse width, time of arrival, bearing and signal strength.

The data used by the ATAC Simulater consists of four integers for each pulse. These words are:

1. Frequency bin. The bins are numbered in ascending order of frequency from 1-128 although the only restriction

is that there has to be less than a total of 255 bins for the current ATAC Simulater program and the 0 bin cannot be used as it is used by ATAC to signal a no data state.

2. Pulse width. The pulse width word is an integer in the range 0-65,535. Any time scaling factor may be used. The current convention adopted is that the integer 1 corresponds to a time interval of 0.1 microseconds. This allows representation of any pulse width from 0-6.5 milliseconds to within 0.05 microseconds.

3. Time of arrival. Time of arrival is expressed as a double precision (32 bit) integer. Word 3 is the least significant half of this enteger and holds time values up to 0.84 seconds with the least significant bit corresponding to 25.6 microseconds.

4. Time of arrival. The upper half of the time word holds time values up to 15.3 hours. This clock update rate and time measurement accuracy is considered adequate for contact analysis and PRI measurements.

The DSP generates data in the above format at a rate of approximately 200 pulses per second.

B. PRINCIPLES OF OPERATION

1. Position information. All position information is initially supplied in X-Y grid coordinates in nautical miles. Velocities are input in knots and direction of travel in degrees clockwise from North. The DSP converts this information to X-Y coordinates in yards relative to the IFM and

relative velocity components in the X and Y directions in yards per second. The relative velocity, Closest Point of Approach (CPA) and distance to CPA is calculated for all platforms.

2. Emitter input data consists of the transmit frequency in MHZ, pulse width in microseconds, PRI in milliseconds, transmitted power in KW, antenna gain in db, antenna beam width in degrees and antenna scan rate in seconds per revolution. DSP converts all input units to common units for computation.

3. IFM input data consists of the maximum and minimum frequencies of its pass band in MHZ, the minimum measurement time of the IFM in microseconds and the threshhold sensitivity in db.

DSP utilizes the input data to calculate the received signal strength for each emitter as a function of range and the angle between the emitters antenna and the IFM platform. The maximum range of detection assuming maximum antenna gain is calculated and compared with the initial range and the CPA to determine when the emitter will come within range of the IFM, if ever. The pulse repetition cycle and antenna initial angle are assigned by assuming a uniform distribution of starting positions in these cycles and accessing a uniform distribution random number generator. When the emitter is within range, the portions of the antenna scan cycle that the antenna gain is high enough to allow detection are calculated.

On the basis of the above calculations, DSP stores the time of arrival at the IFM, counting transit times, for the first pulse of each emitter which will be received with a signal strength equal to or greater than the receiver sensitivity. These times are stored in an array which is then searched to find the next occurring time of arrival. As each pulse time of arrival becomes the next in the sequence of the receiver pulse train, DSP calculates the time of arrival of the next pulse from that emitter and stores that time in the array. Each time update continues to consider the transit antenna scan and the relative motion between the IFM and the emitter platform.

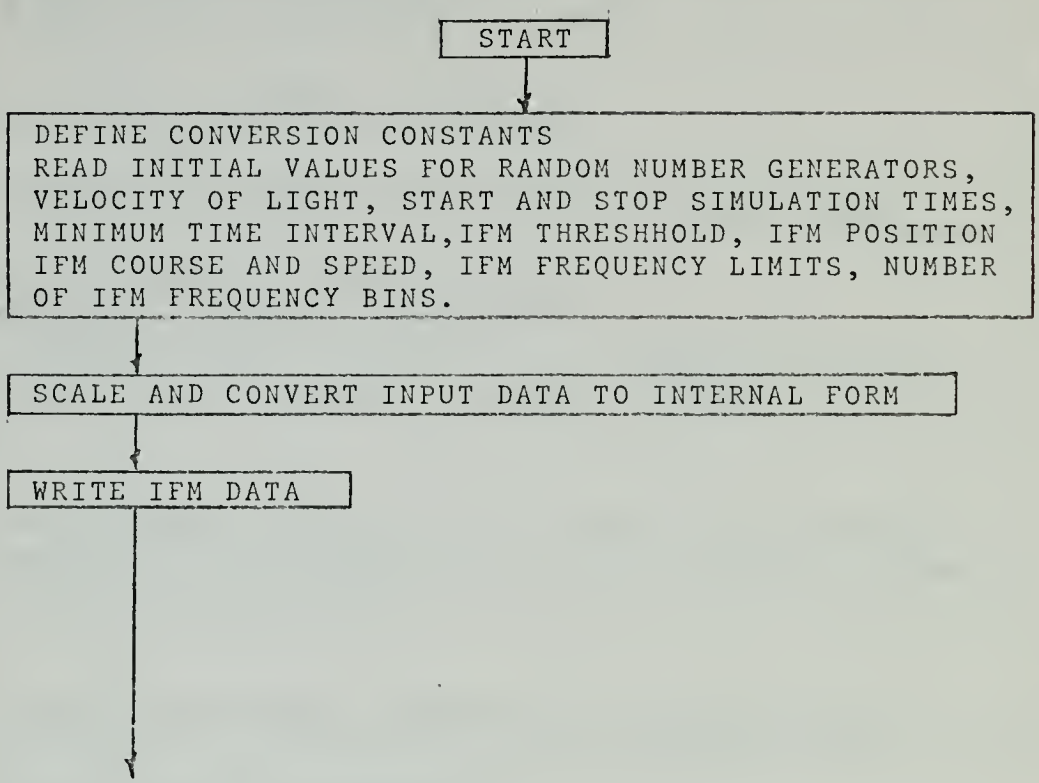
The DSP currently models the IFM with a 100% probability of detection of all pulses arriving with signal strengths equal to or greater than the threshhold sensitivity. Additional coding can readily be added to provide a less than perfect detection scheme, false alarms, and an IFM directional antenna.

The DSP has been provided with selection indicators for selecting random variations to all emitter parameters. Coding has not been included to accomplish any such variation but can be readily added.

The DSP currently includes only two options for the emitter antenna scan type--steady and circular scans. Coding for additional scans can also be readily added.

The overall flow diagram for the DSP is shown in Figure 2.

Data Simulation Program Flow Chart



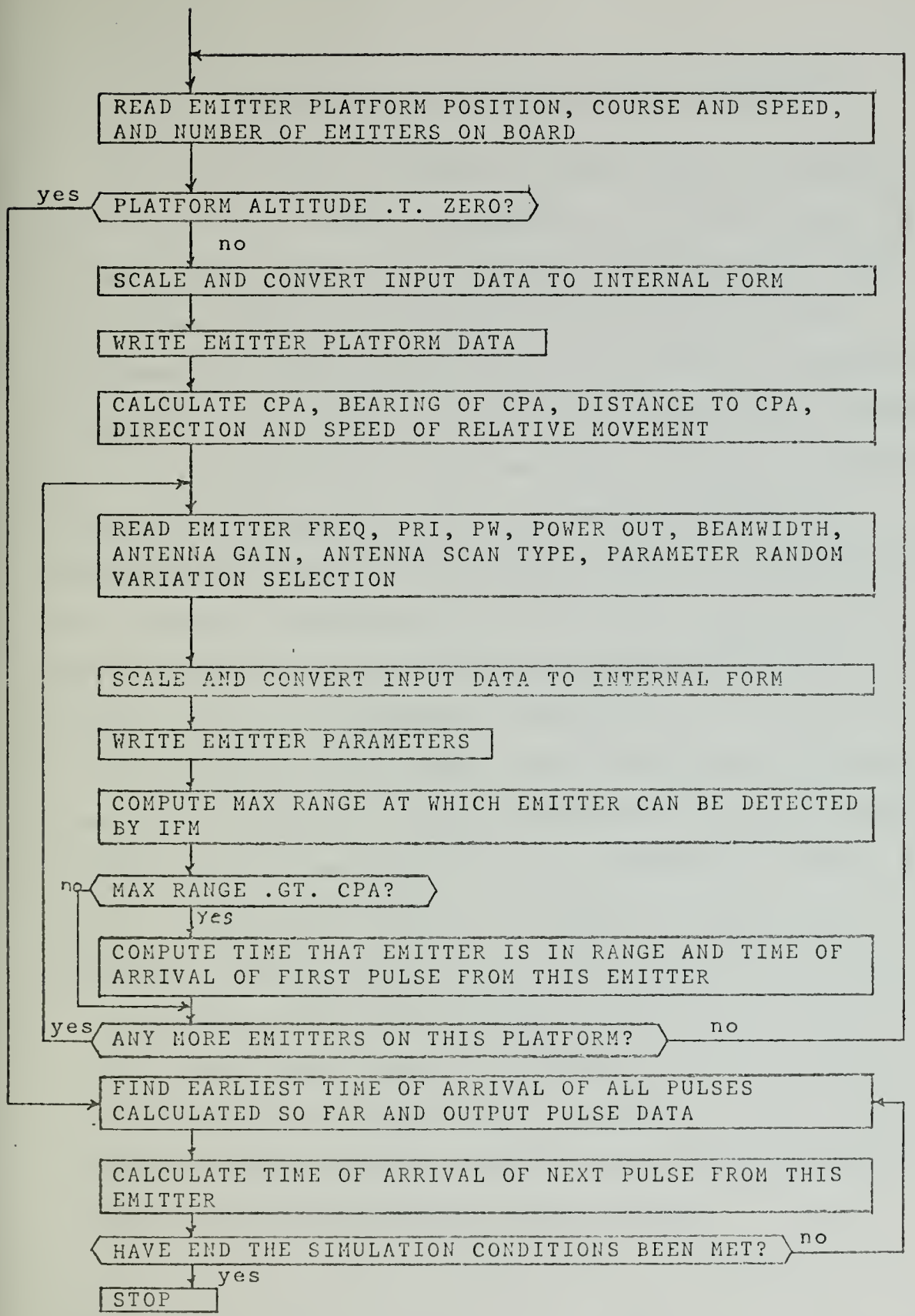



Figure 2. THE DSP ALGORITHM (SIMPLIFIED FLOW CHART)

## VI. OPTIMIZATION OF THE IFM DIGITAL PROCESSOR

The ATAC Assembly Language program listed in Appendix C and described in detail in Appendix D is written using the supplied instruction set. The major advantage of a microprogrammed computer is that the instruction set can be tailored for optimum efficiency in execution of a specific algorithm. Two primary results of this optimization are the saving of instruction fetch times when a series of instructions are microcoded and the time savings realized by performing operations simultaneously when the data paths do not overlap. Modifications to the instruction set are constrained only by the hardware configuration of the computer and the programmer's imagination.

In addition to the time savings associated with modifying the instruction set additional benefits can be accrued by modifying the hardware to allow simultaneous input and output operations, buffering the memory interfaces with high speed registers and instruction interleaving techniques.

Modifications of both microcode and computer hardware can be tested and evaluated by using the APSS Microprogramming Software. By simulating the desired microcode sequences, debugging and optimization efforts can be expended before a commitment to actual design change is made.

A. MICROCODE SEQUENCES

Series of instructions which are shown to use a large percentage of the execution time of the program should be examined for microcoding first. The time expended by each instruction is output by the APSS Instruction Level Simulator in the form of a time histogram. An example of this histogram is shown in Appendix D. This histogram shows that the two instructions which use the most time during the execution of the program are the store data and load new data instructions. The program sequence which utilizes these instructions is required for every input pulse and is listed below.

| Instruction | Purpose | Execution Time (microseconds) |
|---|---|---|
| STK DR,8,4 | Store data | 3.5 |
| BRCS U,TOP | Branch to beginning | .8750 |
| ADD IS,0,4 | Increment address | .4375 |
| RET DR,0,4 | Fetch new data | 3.5 |
| LDR R,5,1 | Move frequency word | .4375 |
| BRCS Z,TOP | Test for presence of data | .4375 |
| LDR IS,1,0 | Put 0 in register 1 | .4375 |
| STR RX,1,0,1 | Store a 0 in the data block | 1.5625 |
| ADD IS,0,4 | Reset address pointer | .4375 |
| LDR DX,6,EF,5 | Fetch frequency file entry | 1.3125 |
| BRCS Z,STAR1 | Test for a pulse train file | .4375 |
| | Total time expended | 13.8125 |

The average time expended per input data set during simulation runs was 45 microseconds. It is seen that 30%

of this time is spent in the store/fetch phase. This sequence placed in microcode would save an initial 4.3 microseconds in instruction fetch times. Additional savings are possible by the simultaneous incrementing of the address registers and simultaneous input and output operations as the input data and output data are using independent segments of memory. It is estimated that the execution time of the microcoded sequence would be approximately 6 microseconds which reduces the program execution time per pulse to approximately 40 microseconds. Additional microcode sequences can be developed, especially for the routine update data section of the program. A time saving of 30% is probable.

Using the technique of microcoding instruction sequences and optimizing the microprogram data flow, the average time per pulse can be reduced to 25 to 30 microseconds. This step alone increases the average pulse density capability of the system to 33K -40K pulses per second.

B. HARDWARE MODIFICATION

The time required to access and store the data can be reduced by a factor of four by altering the means of storing it. If the data is stored in four independent memory modules, parallel transfer of the four words is possible. A hardware modification is required to allow the computer to load four registers in parallel by supplying the additional data paths required.

Supplying an input data buffer and an output data buffer each of four words would allow loading the input buffer during execution of instruction sequences in the middle of the loop and then loading the working registers in parallel in one major cycle. For storing data, the buffer can be loaded in parallel in one major cycle and the data stored in memory from the buffer during execution of the remainder of the program. If two buffers are provided both the fetch and store operations can be conducted simultaneously. From the program's point of view, the execution time required for the 13.8 microsecond sequence of V.A above becomes .4375 microseconds.

## VII. FEASIBILITY ANALYSIS OF THE IFM DIGITAL PROCESSOR

Computer simulation of the IFM digital processor has been conducted. The algorithm presented in Chapter III was coded in ATAC Assembly language. The resultant program is described in Appendices C and D. The Data Simulation Program (DSP) was used to provide a realistic environment of received pulses in order to test the concepts presented in Chapters II and III. Sufficient time was not available to conduct simulation runs to test all conditions and to develop more than an initial evaluation of the concept's viability. A summary of results obtained, conclusions and recommendations for further study are included below.

### A. SUMMARY OF RESULTS

The ATAC computer was modelled as provided for in the APSS Instruction Level Simulator using the program described in detail in Appendix C. Due to limitations imposed by the Simulator, data was inserted into the ATAC constructive memory prior to the beginning of simulation. The data was prepared by the Data Simulation Program described in detail in Appendix D. A total of 1024 radar pulses were utilized in order to check on the accuracy of pulse train building, the development of PRI information and the average time required to handle each pulse. The results of the simulation were the completely accurate discrimination of pulses by frequency and pulse width and the calculation of

the correct PRI in all cases. The total time required for the ATAC to process 1024 pulses was 0.046 seconds or an average of 45 microseconds per pulse. The results are summarized in the table below.

| | |
|---|---|
| Simulation input data. | |
| Number of radar emitters | 9 |
| Total number of pulses utilized | 1024 |
| Total real time simulated | 0.1568 sec |
| Simulation results. | |
| Accuracy of pulse association | 100% |
| Total ATAC simulated computer time | 0.046 sec |
| Average time per pulse | 45 microsec |
| Maximum average data rate possible | 22,000 pps |

## B. CONCLUSIONS

The simulation was conducted with stable pulse widths and PRI. Even so, on the basis of the results of the simulation and on the information contained in Chapter V, it is feasible to construct a system capable of significant amounts of automatic signal classification in a high density environment with the average rate of information in the bandwidth of an IFM in the range of 25K to 50K pulses per second.

## C. RECOMMENDATIONS

Based on the results of this study, the following recommendations for further study are submitted.

1. Continue simulating the high density environment with the programs listed in Appendices C and D adding to the complexity of the input data in order to determine the optimum algorithm for the solution of the problem.

2. Use the results of the simulations to develop additional special purpose instructions which optimize execution of the programmed algorithm.

3. Use the results of the simulations to develop desirable hardware reconfigurations which will optimize the execution of the programmed algorithm.

4. Use the features provided in APSS to test and evaluate the newly developed hardware configuration and instruction set.

5. Expand the basic procedures outlined herein to other fields that encounter high data rates and require automatic data processing. Examples are radar systems, communications systems and sonar systems. The desirability of specifically designed processing equipment has been demonstrated. Current technology is producing these processors which eliminate the processing time and price penalties paid when utilizing general purpose digital computers.

# APPENDIX A

## INSTALLATION OF THE ATAC PROGRAMMING SUPPORT SYSTEM (APSS) AT THE NAVAL POSTGRADUATE SCHOOL

--The ATAC Programming Support System operates on a local or remote IBM 360/370...

ATAC Principles of Operation (Volume 1)

--If something can possibly go wrong, it will...

Murphy's First Law of Dynamics

A. Installation of the ATAC Programming Support System on the IBM 360/67 Operating System at the Naval Postgraduate School by the author was required in order to allow programming of algorithms in an actual representative machine assembly language and to determine execution times and overload points for the algorithm/computer team in an EW environment. After negotiations with ATI Corporation, arrangements were made for a tape copy of the portions of APSS required to conduct instruction level simulations. This tape was received on 10 May 1974. The ensuing events are presented as an example of interfacing problems that can be encountered. Problems encountered during the program implementation are listed in the order of their occurrance along with the corrective action taken.

1. Tape density.

a. The original tape was prepared by ATI using a remote terminal to an IBM 370 installation. This tape was

taken to NPS by the author and an attempt was made to copy from the tape. The tape simply ran out indicating that the Operating System was unable to find any of the specified files on the tape. A utility program 'TAPEOUT' was used to determine what, if anything was on the tape. According to the utility program output there was nothing understandable recorded on the tape at all. A belated investigation of the JCL instructions that produced the tape revealed that it had been prepared on a dual density tape unit and recorded with a density of 1600 bits per inch. The installation at NPS does not support recording densities of greater than 800 bpi.

b. Corrective action. A second tape was generated by ATI specifically recorded at 800 bpi. This tape was picked up by the author on 14 May 1974.

2. Block length.

a. Files 1 and 2 of the new tape were successfully read by the NPS computer. These files contained instructions on how to access File 3 which contained the Instruction Level Simulator in object module form. The required steps were:

1) Copy the object modules from the tape onto a direct access storage device such as a disk pack.

2) Build an absolute load module by providing local linking data between the various portions of the program contained in the object modules. In object module form, the program consists of a group of subroutines which

are stored with instruction and data locations specified as relative to the individual subroutines start point. Building a load module orders these individual sub programs and establishes a common relative base address so that when this entry point is specified (when loading the program for execution) the instruction sequence proceeds as originally designed, including linking between these sub programs.

3) Load and execute the APSS Simulator.

Step 1 was accomplished. However, when attempting to access the file created by copying the tape, the computer was unable to identify the contents of the file and returned the error message 'Permanent I/O Error on Attempting to read File Headings'. Investigation indicated that this could be caused by an incompatible block length of the original object modules. Due to size limitations and buffering considerations, records (data) transferred by computers are organized into blocks of data. These blocks are identified by length and where the next sequential block of data is to be found. To make sense to a computer, when the data consists of an executable program, it must be reassembled with the same block structure in the proper alignment that it had when the program was assembled. Investigation revealed that the original assembly of the program was conducted using a 3330 disk pack and was created with a block length of 13K bytes. The maximum block length supportable at NPS is 7K bytes using a 2314 disk pack. This meant that the NPS

computer could not find the beginning of the second and subsequent data blocks containing the program.

b. Corrective action. When finally diagnosed, this problem was corrected by specifying a track overflow option. This option allows data blocks to 'spill over' onto adjacent tracks of the 2314 disk pack. Use of this feature is dangerous in that the data overflows onto the adjacent track no matter who the adjacent track belongs to or what it contains. This requires using a disk pack with enough room on it so that the entire program can be assigned contiguous tracks.

3. Incompatible instructions.

a. By using the track overflow option, the program was successfully copied into temporary direct access storage and a load module was created using standard IBM supplied utility programs. At this point the program was in executable form. The first attempt to use the program was made using a bench mark program supplied by ATI which was to be used to ensure proper operation. When this program was loaded, the object deck was assembled properly. However, when the simulation itself was attempted, the NPS computer promptly quit and listed an error message which in idiomatic translation means 'I don't understand what you are telling me to do and so I quit'. A dump of the core locations containing the offending instruction revealed the presence of a number of instructions which were implemented for the IBM 370 and are not executable (understandable to) by an IBM 360.

Communication with ATI disclosed that the affected portion of the Simulation Program was written in IBM 370 Assembly Language and that there were ten of these instructions. After further investigation by the author, a similar 360 instruction was selected that was presumed to operate properly for the majority of cases. This new instruction word was placed in the ten locations using a local utility program. On completion, the bench mark program was successfully executed and comparison with the version run at ATI showed that identical results were obtained.

# APPENDIX B

## INSTRUCTIONS FOR USE OF THE APSS INSTRUCTION LEVEL SIMULATOR ON THE IBM 360/67 OPERATING SYSTEM AT THE NAVAL POSTGRADUATE SCHOOL

A. For general and detailed information on how to use the IBM 360/67 OS, the reader is referred to the W. R. CHURCH COMPUTER CENTER USER'S MANUAL.

The ATAC Programming Support System Instruction Level Simulator currently resides in load module form on Cell 5 of the 2321 Data Cell under file name S1434.PDF.APSS. The steps recommended for use of this program are:

1. Write any desired program in ATAC Assembly Lanaguage following the directions of ATAC PRINCIPLES OF OPERATION (VOLUME I).

2. Include desired simulated Input/Output operations following the directions of ATAC APSS INSTRUCTION SIMULATOR (VOLUME 2, Section IV). For this purpose it may be necessary to replace supplied FORTRAN dummy subroutines with subroutines written to simulate data generation and interrupt timing. The Job Control Language sequence required to accomplish this replacement is shown in paragraph C.

3. Include desired Simulator Control instructions following the directions of ATAC APSS INSTRUCTION SIMULATOR (VOLUME 2, Section IV).

4. Prepare the coded version of 1, 2, and 3 above following the directions of ATAC APSS ASSEMBLER/LOADER (VOLUME 2, Section III) for precise card formats.

5. Construct the input card deck for the desired job including required control features in accordance with ATAC APSS JOB MANAGEMENT (VOLUME 2, Section I).

On completion of 1 through 5, the user will have a card deck the first card of which is $JOB and the last card is $END. The following IBM JCL cards are required in order to execute the program.

```
// (STANDARD JOB CARD.  SEE USER'S MANUAL)
// TEST EXEC PGM=APPS,REGION=250K
// STEPLIB DD DSN=S1434.PDF.APSS,UNIT=2321,
//      DISP=(OLD,KEEP),VOL=SER=CEL005,DCB=BLKSIZE=400
// FT06F001 DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,
//      BLKSIZE=3325)
// FT05F001 DD DDNAME=SYSIN
// FT08F001 DD UNIT=SYSDA,SPACE=(CYL,1)
// FT09F001 DD UNIT=SYSDA,SPACE=(CYL,(7,2)),
//      DCB=(RECFM=VBS,BLKSIZE=7180,LRECL=92)
// FT10F001 DD UNIT=SYSDA,SPACE=(CYL,(7,2)),
//      DCB=(RECFM=VBS,BLKSIZE=4204,LRECL=42,BUFNO=1)
// FT20F001 DD UNIT=SYSDA,SPACE=(CYL,(7,2)),
//      DCB=(BUFNO=1,RECFM=VBS,BLKSIZE=2004,LRECL=500)
// SYSPRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,
//      BLKSIZE=3325)
// SYSIN DD *

$JOB

     .
     .
     .
     APSS PROGRAM
     .
     .
     .
$END
/*
```

B. The source tape for the APSS Instruction level simulator is stored at the W. R. CHURCH COMPUTER CENTER under the name 'ATI.105'. Files 1 and 2 on this tape contain instructions for the use of the entire APSS system. The instructions given are not in general applicable for use and retrieving these files is not considered desirable. If retrieval is warranted, use the IBM utility program IEBGENER. For further information refer to the USER'S MANUAL.

File 3 contains the object module form of the APSS Instruction Level Simulator. The following steps are required to prepare it for use in the event of failure to the Data Cell. All of these steps may be accomplished in one JOB if desired. The total CPU time required is approximately 10 seconds.

1. Move the object modules from the tape to disk storage. ENSURE the actual disk pack used has sufficient unallocated room for contiguous assignment of tracks. The object module file has a block length of 13K bytes and the track overflow feature has to be selected in order to be able to access the new disk file successfully. The track overflow feature allows data blocks to overflow from one track onto the adjacent track, whether it is assigned to you or not. The normally used USER disks will not usually have sufficient space available. Special arrangements should be made before the file is moved. The required JCL statements are as follows.

```
    // (STANDARD JOB CARD)
    // EXEC PGM=IEHMOVE,REGION=150K
    // SYSPRINT DD SYSOUT=A
    // SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(40),,CONTIG)
(1) // DA1 DD UNIT=2314,DSN=S1434.ATI.PDF1,SPACE=(TRK,
    //          (80,10,10),,CONTIG),
(2) //          DCB=RECFM=T,
    //          DISP=(NEW,KEEP),VCL=SER=PAT003,
    //          LABEL=RETPD=060
    // T1TAPE DD UNIT=(2400,,DEFER),DISP=NEW,PASS),
    //          LABEL=(3,SL,,IN), DCB=(DEN=2,BLKSIZE=800,
    //          LRECL=80,RECFM=FB),VOL=SER=ATI105,
    //          DSN=ATI.TEMP.APSS1
    // SYSIN DD *
(3)    COPY PDS=ATI.TEMP.APSS1,TO=2314=PAT003,FROMDD=T1TAPE,
             TODD=DA1, FROM=2400=(ATI105,3),
             RENAME=S1434.ATI.PDF1
    / *
```

Note 1: File names selected must be consistent throughout.

Note 2: This selects the track overflow option.

Note 3: The asterisk (*) in column 72 is required as an indication that another card follows which is a continuation of the COPY PDS card.

2. The object modules now on the disk must be built into a load module and stored in a permanent file. The Data Cell is adequate for this purpose. The following JCL cards will accomplish this step.

```
    // BUILD EXEC PGM=IEWL,REGION=100K,PARM='OVLY,
    //      XREF,LET,LIST'
    // SYSPRINT DD SYSOUT=A
    // LIBRARY DD DSN=S1434.ATI.PDF1,UNIT=2314,
    //      VOL=SER=PAT003,DISP=SHR
    // SYSLIB DD DSNAME=SYS1.FORTLIB,DISP=SHR
    // SYSLMOD DD DSNAME=S1434.PDF.APSS,UNIT=2321,
    //      VOL=SER=CEL005, DISP=(NEW,KEEP),
    //      LABEL=RETPD=060,
    //      SPACE=(TRK,(300,20,2),RLSE)
```

```
// SYSUT1 DD UNIT=SYSDA,SPACE=(TRK,(19,19),,
//      CONTIG),SEP=SYSLMOD
// SYSLIN DD *
   INCLUDE LIBRARY (PMUDL)
   CHANGE MSIM(IHESAPD)
   INCLUDE LIBRARY(LIBR2)
   INCLUDE LIBRARY(ATACIO)
   INCLUDE LIBRARY(MSIM4,ASEM5,SIMU11,SIMTR1,SIMIO1,GUL)
   INCLUDE LIBRARY(XPLMON)
   OVERLAY A1
   INSERT MSIMUL,*MSIMULA,IHENTRY,IHESAP
   INSERT MINT,IN,OUT
   INSERT IHEDBN,IHEXID
   INSERT IHEBSM,IHECSM
   INSERT IHEBSK, IHEIOX, IHEIOP, IHEDID, IHEDOB
   INSERT IHEDIB, IHEDOA, IHEIOB
   INSERT IHEIOA, IHEOCL
   INSERT IHEBSD, IHEBSF
   INSERT IHEJXS
   INSERT IHEOSD, IHEOST, IHEBSI
   INSERT IHEVPF, IHEDMA, IHEVFB
   INSERT IHEDNC, IHEVFD, IHEVFA, IHEVPD, IHEVPB, IHEVSC
   INSERT IHEVSD, IHEVFE, IHEDCN, IHEUPB
   INSERT IHEVFC, IHEVPE, IHEVPG, IHEVQB, IHEVQC
   INSERT IHEABN, IHEIOD, IHEIOF, IHEPRT, IHEVQA, IHESPRT
   INSERT IHEBEG, IHEERR, IHESIZ
   INSERT MISET
   OVERLAY A1
   INSERT ASSEM, REWIND,REW72,DSKOUT,CARDIN,DISKIN,
        ERPRT,PRIADD
   INSERT WRDATA,PRICOM,PRINOR,WRITEX,REFTIT,PREF,ERTIT
   OVERLAY A1
   INSERT PARMRD, PRESIM
   OVERLAY A1
   INSERT SMLTR,NRMTRM,STRTSM,TPAGE
   INSERT RDCRD,ABNRMT,ARTHFP,TRACE,HGRAM,HGRAMI,HGRAMS
   INSERT IOINIT,ACTIVE,SIMTIM
   INSERT DEVDTA,ACT,PRGTIM,RAND,TIME,INT,RAND,DEADT,DEBUG
   OVERLAY A2
   INSERT LEVEL,IOR,SPLIT,DMAIOI,DMAIOA,DMAIOD,RIOIO
   INSERT REMACT,DMA,DMATM,RIO,RIOTM,RIOINT,INTOLY,DMAINT
   INSERT DTRAN,PUTACT,RANDOM
   OVERLAY A1
   INSERT HGPRNT
   OVERLAY A1
   INSERT LOADER
   OVERLAY A1
   INSERT LINK,ENTEXT,SLLH
   OVERLAY A1
   INSERT PLATAC,IOPACK
   OVERLAY $OBJECT(REGION)
```

```
    INSERT OBJECT,INIT,LIB,RCALPH,RCHEX,RCINT
    INSERT MDATE
    OVERLAY $DUMP(REGION)
    INSERT SMDUMP,PAGE
    ENTRY MAIN
         NAME APSS
/ *
```

3. When the load modules have been built onto the Data Cell, the disk file is no longer required and should be scratched. The following JCL, cards will accomplish this.

```
// EXEC PGM=IEHPROGM
// DD1 DD DISP=(OLD,DELETE),UNIT-2314,VOL=SER=PAT003
// SYSPRINT DD SYSOUT=A
// STSIN DD *
   SCRATCH DSNAME=S1434.ATI.PDF1
   / *
```

4. At this point, the load module is on the Data Cell but there are 11 memory locations which are required to be individually changed in order for the program to execute. See APPENDIX A for detailed reasons why. Changing the memory locations requires the use of a utility program which is under close control of the Computer Center personnel for obvious reasons. The locations, current contents and required new contents are listed below. For assistance in making this change, see the Director of Operations at the Computer Center.

All numbers are hexidecimal
NAME APSS SMLTR

| Location | Old Contents | New Contents |
|---|---|---|
| 115E | BD53 | 4950 |
| 1166 | BD53 | 4950 |

| Location | Old Contents | New Contents |
|---|---|---|
| 1176 | BD53 | 4950 |
| 117E | BD53 | 4950 |
| 118E | BD53 | 4950 |
| 1196 | BD53 | 4950 |
| 11A2 | BD53 | 4950 |
| 1200 | BD53 | 4950 |
| 1208 | BD53 | 4950 |
| 1214 | BD53 | 4950 |
| 13B2 | A5EC | B39B |

5. The APSS Instruction Level Simulator is now ready for use.

C. A subroutine may be substituted for any of the dummy Input/Output subroutines supplied very simply. The first step is to write and debug the new subroutine. Care must be taken to ensure that the new subroutine has the same name and identical calling arguments as the old one. Once the subroutine is fully ready for use, it may replace the original one as shown below. It must be pointed out that once this step is accomplished, the only way to retrieve the original subroutine is to build the entire system over again.

```
// (STANDARD JOB CARD)
// EXEC FORTCL
// FORT.SYSIN DD *
        (FORTRAN SOURCE DECK)
/ *
```

```
// LINK.SYSLMOD DD DSNAME=S1434.PDF.PASS,DISP=(OLD,KEEP),
// UNIT =2321,VOL=SER=CEL005
// LINK.SYSIN DD *
   NAME (SUBROUTINE NAME) (R)
```

# APPENDIX C

## THE IFM DIGITAL PROCESSOR PROGRAM LISTING

The IFM Digital Processor Program (IDPP) is the implementation of the algorithm presented in Chapter III in ATAC Assembly Language. For ease in examination of the coding used, a listing of the instructions used in this program and their functions is included in Appendix E.

1. Program Listing. The program is listed by the APSS Assembler and provides the following information.

a. Column 1. The first column lists the instruction location relative to the start of the program in hexadecimal. The program actual entry address is location 10 so that in the simulator output listings, locations appear $10_{16}$ higher than they do on the program listing.

b. Columns 2. The second and third columns contain the actual hexidecimal contents of the instruction locations. Single word instructions do not have an entry in the third column.

c. The image of the input program cards is reproduced. Lines that have a period occuring as the first entry are comment cards.

d. The sequence number of the cards is simply the order in which they occurred in the data stream.

2. The program has been written as one entity in order to save linking by subroutines and paying the associated time

overhead. The different functions of the program have been separated by blank comment cards for ease of reference. The basic function of each instruction has been included in the comments made in the margin adjacent to the instruction.

3. Beginning with the location HADRS, the following cards establish data storage requirements and represent the only means of interpreting the results of the simulation. The Simulator automatically dumps the entire ATAC core storage on completion of the simulation. By referencing the locations of the various variables and data, the contents at the completion of the simulation run may be determined. A summary of the meaning of the data storage locations is given below.

| Variable | Interpretation |
|---|---|
| HADRS | Used to specify limits of the instruction versus time Histogram |
| TRACK | The output pulse train file. Individual files begin every $10_{16}$ locations. Interpretation of the word values is contained in Section III.C.1,2,3. |
| EFILE | Frequency input file. The contents of EFILE locations show the mapping conducted from frequency information to the proper pulse trains. |
| TFILE | The contact file entry file. Locations correspond to the pulse train file and are in the same order. The contents of even |

| Variable | Interpretation |
|---|---|
| | words of the file contain the entry location of the corresponding pulse train file. |
| OFLO | The overflow storage area. |
| OFLOW | Variable used to point to the next available location in the overflow area. |
| NEXTF | Variable used to point to the last pulse train file activated. |
| PFILE | Used to define the extent of TFILE for the purposes of initializing the TFILE locations to the proper value by reading in data cards. |
| DATA | Locations used to store input pulse data. |
| DATAR | Used to define the extent of DATA for the purposes of initializing the DATA locations to the data values by reading in data cards. |

4. The simulator output.

a. A print of the contents of each of the 16 general registers, the condition code register, the input/output interrupt masks and the simulated computer time elapsed since the beginning of the simulation can be provided. This print out must be specifically requested with a Trace request. It is turned on and off within the program to avoid a line of output for each and every instruction executed by the simulated computer. The feature is very useful for checking the program flow.

b. The simulation histogram of time expended by the instructions at each real location (10 + the program listing location). This feature shows which instructions take more time in execution than others. By noting which instructions or instruction sequences take abnormal amounts of time, the program flow can be changed to improve the overall efficiency. Additionally this output shows those instructions which should be looked at with the possibility of redefining a new microcode instruction or modifying the hardware configuration in order to improve the time required for execution of the program.

c. Core dump. This output is the primary means of verifying proper operation of the program. Variable locations in the program listing are given in actual core locations if the variable is in Random Access Memory (RAM) or Read Mostly Memory (RMM).

```
LOC   OBJECT CODE     CARD IMAGE                                                                  CARDNUM
                            IDT   IFM                                                          IFM0000     1
                            ENTRY   TOP                                                        IFM0010     2
                            ORG   ROM                                                          IFM0020     3
0000  FE05 0000             SIM   5,0                  TURN OFF THE TRACE                      IFM0030     4
                      .                                                                        IFM0040     5
                      .                                                                        IFM0050     6
                      .INFORMATION FILTER PROGRAM FOR THE OUTPUT OF AN IFM RECEIVER.  THE      IFM0060     7
                      .RECEIVER LOADS DIRECTLY INTO A MEMORY BLOCK A SERIES OF 4 WORDS FOR     IFM0070     8
                      .EACH PULSE RECEIVED.                                                    IFM0080     9
                      .       1.  FREQUENCY  -  IN TERMS OF THE FREQUENCY BIN OF THE PULSE.    IFM0090    10
                      .       2.  PULSE WIDTH  -  16 BIT INTEGER.  LSB = 0.1 MICROSECOND.      IFM0100    11
                      .       3.  TIME DOUBLE PRECISION INTEGER.  LSB = 25.6 MICROSEC          IFM0110    12
                      .       4.  MOST SIGNIFICANT HALF OF THE TIME DATA WORD                  IFM0120    13
                      .THE FILTER PROGRAM RETRIEVES THIS INFORMATION FROM THE MEMORY BLOCK     IFM0130    14
                      .AND ASSOCIATES IT WITH PULSE TRAINS, DETERMINES A SIMPLE PULSE REP      IFM0140    15
                      .INTERVAL, IF ONE EXISTS, AND OUTPUTS PULSE TRAIN INFORMATION TO A       IFM0150    16
                      .CENTRAL PROCESSOR.  THE OUTPUT ACTION CONSISTS OF STORING THE PULSE     IFM0160    17
                      .TRAIN INFORMATION IN ANOTHER MEMORY BLOCK TO WHICH THE CENTRAL          IFM0170    18
                      .PROCESSOR CAN ACCESS DIRECTLY.  THE ONLY OPERATIONS THAT REQUIRE        IFM0180    19
                      .INTERUPTS TO EITHER THE INFORMATION FILTER OR THE CENTRAL PROCESSOR      IFM0190    20
                      .IS THE PASSING OF CONTROL ORDERS FROM THE CENTRAL PROCESSOR             IFM0200    21
                      .TO THE INFORMATION FILTER.                                              IFM0210    22
                      .                                                                        IFM0220    23
                      .                                                                        IFM0230    24
                      .REGISTER ASSIGNMENTS:                                                   IFM0240    25
                      .       REG 0  INPUT DATA FILE POINTER          REG  8  PULSE TRAIN FIL  IFM0250    26
                      .       REG 1  FREQUENCY BIN; ZERO              REG  9  FILE PULSE WIDT  IFM0260    27
                      .       REG 2  TIME OF ARRIVAL                  REG 10  FILE PRI         IFM0270    28
                      .       REG 3  TIME OF ARRIVAL                  REG 11  FILE TIME        IFM0280    29
                      .       REG 4  PULSE WIDTH                      REG 12  FILE TIME        IFM0290    30
                      .       REG 5  WORKING                          REG 13  WORKING          IFM0300    31
                      .       REG 6  PULSE TRAIN DIRECTORY            REG 14  WORKING          IFM0310    32
                      .       REG 7  PULSE TRAIN FILE POINTER REL ADD REG 15  RETURN ADDRESS    IFM0320    33
                      .                                                                        IFM0330    34
                      .                                                                        IFM0340    35
                      .                                                                        IFM0350    36
                      .SIMULATION CONTROL SET UP                                               IFM0360    37
                      .                                                                        IFM0370    38
0002  FE09 0003             SIM   9,HADRS              SET UP INST VS TIME HISTOGRAM           IFM0380    39
0004  FE01 1A02             SIM   1,PFILE              LOAD CONTACT FILE LINK ADDRESSES        IFM0390    40
0006  FE01 1E04             SIM   1,DATAR              LOAD DATA                               IFM0400    41
0008  E100 1A04             LDR   I,0,DATA             SET INPUT DATA FILE POINTER             IFM0410    42
000A  E10F FC00             LDR   I,15,-1024           SET PULSE COUNTER                       IFM0420    43
000C  E101 0080             LDR   I,1,128              INITIALIZE OVERFLOW POINTER             IFM0430    44
000E  9C01 1980             STR   0,1,OFLOW            STORE OVERFLOW POINTER                  IFM0440    45
0010  0037                 BRCS  U,TOP                GO GET FIRST PULSE                      IFM0450    46
0011  FE00 0000  DONE       SIM   0,0                  TURN ON TRACE TO GET FINAL TIME         IFM0460    47
0013  FE07 0000             SIM   7,0                  STOP SIMULATION                         IFM0470    48
                      .                                                                         IFM480    49
                      .                                                                         IFM490    50
                      .                                                                        IFM0500    51
                      .                                                                        IFM0510    52
                      .                                                                        IFM0520    53
                      .       THE ACTUAL IFM PROGRAM FOLLOWS                                   IFM0530    54
```

```
LOC   OBJECT CODE     CARD IMAGE                                                    CARDNUM
                      *                                                             IFM0540    55
                      *                                                             IFM0550    56
                      *                                                             IFM0560    57
0015  6040            TDP     ADD   IS,D,4      SET DATA FILE POINTER FOR THIS BLOCK IFM0570    58
0016  601F                    AOD   IS,15,1                                         IFM0580    59
0017  0F62                    BRCS  Z,DONE                                        IFM0590    60
0018  8E30                    RET   DR,D,4       FETCH 4 WORDS OF PULSE INFORMATION IFM0600    61
0019  E015                    LDR   R,5,1        MOVE FREQ BIN ADDRESS AND SET CCM IFM0610    62
001A  0F92                    BRCS  Z,TOP        BRANCH TO TOP IF NO DATA STORED   IFM0620    63
001B  4001                    LOR   IS,1,0       DATA AVAILABLE.  RESET DATA BLOCK IFM0630    64
001C  1101                    STR   RX,1,D,1     PUT A ZERO IN THE FREQ LOCATION   IFM0640    65
001D  6040                    ADD   IS,D,4       SET DATA FILE POINTER FOR NEXT BLOCK IFM0650  66
001E  E057                    LOR   R,7,5        SET RELATIVE ADDRESS OF FREQ FILE IFM0660    67
001F  E376  1800      FLINK   LDR   DX,6,EFILE,7 PUT PULSE TRAIN DIRECTORY INTO REG 6 IFM0670 68
0021  0222                    BRCS  Z,START      IF NO FILE, GO TO START ONE       IFM0680    69
0022  C46E                    LDL8  R,14,6       LOAD PT FILE REL ADDRESS          IFM0690    70
0023  E3E8  1981              LOR   DX,8,TFILE,14 PT FILE ENTRY ADD IN REG 8       IFM0700    71
0025  5819                    LOR   RX,9,1,8     GET FIRST WORD OF PULSE TRAIN FILE IFM0710    72
0026  E090                    LDR   R,13,9       COPY INTO WORKING REG             IFM0720    73
0027  844D                    SUB   R,13,4       FIND DIFFERENCE BETWEEN PW'S      IFM0730    74
0028  C001                    BRCS  P,TEST       IF NEW .LT. DLO, GO TEST          IFM0740    75
0029  90D0                    LAC   R,13,13      FIND MAGNITUDE                    IFM0750    76
002A  2040            TEST    CMP   IS,13,4      COMPARE MAGNITUDE TO 0.4 MICROSECS IFM0760   77
002B  0301                    BRCS  P,NTRY       IF PW OUT OF LIMITS, TRY NEXT FILE IFM0770   78
002C  6011                    AOD   IS,1,1       THIS IS THE FILE                  IFM0780    79
002D  581A                    LDR   RX,10,1,8    GET NEXT FILE WORD (PRI)          IFM0790    80
002E  03C4                    BRCS  N,BUILD      IF PRI .LT. D, FILE BUILDING      IFM0800    81
002F  601I            UPDATE  AOD   IS,1,1       GET REST OF FILE                  IFM0810    82
0030  5818                    LDR   RX,11,1,8                                      IFM0820    83
0031  6011                    AOD   IS,1,1                                         IFM0830    84
0032  581C                    LDR   RX,12,1,8    PT FILE NOW IN REG 9-12           IFM0840    85
0033  E026                    LDR   R,6,2        COPY NEW TOA INTO WORKING REG     IFM0850    86
0034  E037                    LDR   R,7,3                                          IFM0860    87
0035  AA86                    DSUB  R,6,11       COMPUTE TIME DIFFERENCE           IFM0870    88
0036  0004                    BRCS  N,TOP        IF DIFF NEG, PULSE OUT OF SEQ     IFM0880    89
0037  0705                    BRCS  NZ,SPR       IF TIME DIF .GT. .8 SECONDS BR SPR IFM0890   90
0038  D460                    LDD8  R,13,6       TEST BEST OF TIME DIFF            IFM0900    91
0039  210D                    CMP   IS,13,16                                       IFM0910    92
003A  0601                    BRCS  P,SPR        IF TIME DIFF .GT. .1 SEC, BR SPR  IFM0920    93
003B  84A6                    SUB   R,6,10       UPDATE PRI INFO                   IFM0930    94
003C  C781                    BRCS  P,INCR       IF NEW .GT. DLO, INCREASE DLD     IFM0940    95
003D  07C4                    BRCS  N,DECR       IF NEW .LT. DLO, DECREASE DLO     IFM0950    96
003E  E028            STORE   LOR   R,11,2       LOAD NEW TIME                     IFM0960    97
003F  E03C                    LOR   R,12,3                                         IFM0970    98
0040  8494                    SUB   R,4,9        UPDATE PW                         IFM0980    99
0041  0781                    BRCS  P,INCP       IF NEW .GT. DLO, INCREASE DLO     IFM0990   100
0042  07C4                    BRCS  N,DECP       IF NEW .LT. DLD, DECREASE DLO     IFM1000   101
0043  8038            STACK   STK   OR,8,4       OUT UPDATED VALUES INTO FILE      IFM1010   102
0044  0CF7                    BRCS  U,TOP        GO BACK TO GET NEXT PULSE         IFM1020   103
                      *                                                             IFM1030   104
                      *                                                             IFM1040   105
                      *                                                             IFM1050   106
0045  E1C0  FF3F      START   LOR   I,13,-193    SET WRAP AROUND MASK              IFM1060   107
```

```
LOC   OBJECT CODE      CARD IMAGE                                                                  CARDNUM
0047  E206 1A01        LDR   D,6,NEXTF      GET LAST FILE USED                                    IFM1070   108
0049  6026      ANOT   ADD   IS,6,2         SET VALUE FOR NEXT FILE                               IFM1080   109
004A  A006             AND   R,6,13         NEXT FILE ADDR                                        IFM1090   110
004B  E368 1981        LDR   DX,8,TFILE,6   GET FILE ENTRY ADDRESS                               IFM1100   111
004D  0FA5             BRCS  NZ,ANOT        IF FILE BUSY, GET THE NEXT ONE                        IFM1110   112
004E  9C06 1A01        STR   0,6,NEXTF      STORE FILE ADDR FOR NEXT TIME                         IFM1120   113
0050  9B65 19B1        STR   DX,5,TFILE,6   STORE FREQUENCY IN CONTACT FILE                       IFM1130   114
0052  6016             ADD   IS,6,1                                                               IFM1140   115
0053  E368 19B1        LDR   DX,B,TFILE,6   LOAD PT FILE ENTRY ADD                               IFM1150   116
0055  9B76 1800        STR   DX,6,EFILE,7   STORE PT FILE LINK                                    IFM1160   117
0057  E049             LDR   R,9,4          PUT INITIAL FILE VALUES IN REG 9-12                   IFM1170   118
0058  4F0A             LDR   IS,10,-16      SET PULSE CTR VALUE IN PRI WORD                       IFM1180   119
0059  E02B             LDR   R,11,2         LOAD TOA                                              IFM1190   120
005A  E03C             LDR   R,12,3                                                               IFM1200   121
005B  B038             STR   DR,8,4         STORE FILE ENTRY                                      IFM1210   122
005C  0B77             BRCS  U,TOP          GO GET NEXT PULSE                                     IFM1220   123
                         .                                                                        IFM1230   124
                         .                                                                        IFM1240   125
                         .                                                                        IFM1250   126
005D  046E        NTRY LDUB  R,14,6         GET THE ADDRESS OF THE NEXT FILE                      IFM1260   127
005E  0012             BRCS  Z,STAR2        IF NO NEXT FILE, START ONE                            IFM1270   128
005F  E0E7             LDR   R,7,14         SET REL ADDRESS OF NEXT FILE                          IFM1280   129
0060  0BD7             BRCS  U,FLINK        GO TRY THE NEXT FILE                                  IFM1290   130
                         .                                                                        IFM1300   131
                         .                                                                        IFM1310   132
                         .                                                                        IFM1320   133
0061  E2CE 19B0  STAR2  LDR   D,14,OFLOW     FIND THE NEXT OVERFLOW LOCATION                       IFM1330   134
0063  B0E0             XCHB  R,13,14        PREPARE THE LINKING WORD                              IFM1340   135
0064  80C6             ADD   R,6,13         LINKING ACCOMPLISHED                                  IFM1350   136
0065  9B76 1800        STR   DX,6,EFILE,7   STORE LINKING WORD                                    IFM1360   137
0067  E0E7             LDR   R,7,14         SET REL ADDRESS OF NEXT FILE                          IFM1370   138
0068  601E             ADD   IS,14,1        INCREMENT OVERFLOW LOCATION                           IFM1380   139
0069  9C0E 1980        STR   0,14,OFLOW     STORE NEXT OVERFLOW LOCATION                          IFM1390   140
006B  0087             BRCS  U,START        GO START THE REST OF THE FILE                         IFM1400   141
                         .                                                                        IFM1410   142
                         .                                                                        IFM1420   143
                         .                                                                        IFM1430   144
006C  601A        BUILD  ADD   IS,10,1       ADD ONE TO PULSE COUNTER IN REG 10                    IFM1440   145
006D  0102             BRCS  Z,PRI          IF ZERO, BRANCH TO CALCULATE PRI                      IFM1450   146
006E  11BA             STR   RX,10,8,1      STORE PULSE COUNTER                                   IFM1460   147
006F  610A             ADD   IS,10,16       CALCULATE WHERE DATA BELONGS                          IFM1470   148
0070  AE1A             SHS   LL,10,2        DATA'S REL ADDR                                       IFM1480   149
0071  8CA8             ADD   R,B,10         DATAS REAL ADDRESS                                    IFM1490   150
0072  E085             LDR   R,5,8          PUT INTO REG 5                                        IFM1500   151
0073  6FE5             ADD   IS,5,-2        ADDR OF THE PREVIOUS TIME                             IFM1510   152
0074  FB15             LDRM  DR,5,2         FETCH PREVIOUS TIME                                   IFM1520   153
0075  E026             LDR   R,11,2         CALCULATE THE TIME DIFFERENCE                         IFM1530   154
0076  E03C             LDR   R,12,3                                                               IFM1540   155
0077  AA62             OSUB   R,2,6                                                               IFM1550   156
0078  09B4             BRCS  N,TOP          IF PULSE OUT OF SEQ, DISREGARD                        IFM1560   157
0079  0002             BRCS  Z,GOOD         IF TIME DIFFERENCE .GT. .8 SEC                        IFM1570   158
007A  4002             LDR   IS,2,0         ZERO TIME DIFFERENCE                                  IFM1580   159
007B  E02A        GOOD LDR   R,10,2         LOAD DATA(PRI)                                        IFM1590   160
```

```
LOC   OBJECT CODE     CARD IMAGE                                                    CARDNUM
007C  E049            LDR   R,9,4      PW                                         IFM1600  161
007D  8D38            STK   DR,8,4     STORE DATA IN FILE                         IFM1610  162
                .     SIM   5,0        TURN OFF THE TRACE                         IFM1620  163
007E  0957            BRCS  U,TOP      GO GET NEXT PULSE                          IFM1630  164
                      :                                                           IFM1640  165
                      :                                                           IFM1650  166
                      :                                                           IFM1660  167
007F  4F11      PRI   LOR   IS,1,-15    GO TO END OF FILE                          IFM1670  168
0080  6408            ADD   IS,8,64                                               IFM1680  169
0081  8E38            RET   DR,8,4      FETCH ENTIRE PULSE ENTRY                  IFM1690  170
0082  E026            LDR   R,6,2       PUT NEW TIME INTO WORKING REG             IFM1700  171
0083  E037            LDR   R,7,3                                                 IFM1710  172
0084  AAB6            DSU8  R,6,11      CALCULATE TIME DIFFERENCE                 IFM1720  173
0085  08E4            BRCS  N,TOP       IF PULSE OUT OF SEQ, DISREGARD            IFM1730  174
0086  4008            LDR   IS,11,0     LOAD NEW TIME DIFF                        IFM1740  175
0087  E06C            LDR   R,12,6      INTO REG 11 AND 12                        IFM1750  176
0088  8094      LP2   ADD   R,4,9       SUM PULSE WIDTHS                          IFM1760  177
0089  A8A6            OADD  R,6,10      SUM TIME DIFFERENCES                      IFM1770  178
008A  6011            ADD   IS,1,1      STEP COUNTER                              IFM1780  179
008B  00D2            BRCS  Z,PEND      IF COUNTER 0, ALL DATA USED               IFM1790  180
008C  6FE8            ADD   IS,8,-2     SET ADDRESS FOR NEXT PREV PULSE           IFM1800  181
008D  8E18            RET   DR,8,2      FETCH PW AND TIME DIFFERENCE              IFM1810  182
008E  84CA            SUB   R,10,12     COMPARE LAST TIME DIFF                    IFM1820  183
008F  C031            BRCS  P,LP3       IF POSITIVE, TEST AT LP3                  IFM1830  184
0090  2FCA            CMP   IS,10,-4    IF NEG, COMPARE WITH -4                   IFM1840  185
0091  0054            BRCS  N,NG        IF .LT. -4, THROW OUT TIME DIFF           IFM1850  186
0092  80CA            AOD   R,10,12     RESET DATA                                IFM1860  187
0093  0F37            BRCS  U,LP2       GO SUM DATA                               IFM1870  188
0094  204A      LP3   CMP   IS,10,4     IF TIME DIFF +, COMPARE WITH +4           IFM1880  189
0095  0011            BRCS  P,NG        IF .GT. 4, THROW OUT TIME DIFF            IFM1890  190
0096  80CA            AOD   R,10,12     RESET DATA                                IFM1900  191
0097  CEF7            BRCS  U,LP2       GO SUM DATA                               IFM1910  192
0098  E0CA      NG    LDR   R,10,12     IF TIME DIFF BAD, USE STANDARD            IFM1920  193
0099  0E07            BRCS  U,LP2       GO SUM DATA                               IFM1930  194
                      :                                                           IFM1940  195
                      :                                                           IFM1950  196
                      :                                                           IFM1960  197
009A  D836      PEND  SHD   RA,6,4      DIVIDE BY 16 TO GET AVG PRI               IFM1970  198
009B  AC34            SHS   RA,4,4      DIVIDE BY 16 TO GET AVG PW                IFM1980  199
009C  8E18            RET   DR,8,2      GET TOA OF FIRST PULSE OF TRAIN           IFM1990  200
009D  6028            ADD   IS,8,2      SET ADDRESS FOR NEXT DATA BLOCK           IFM2000  201
009E  400B            LDR   IS,11,0     SET SPR = 0                               IFM2010  202
009F  400C            LDR   IS,12,0                                               IFM2020  203
00A0  8D38            STK   DR,8,4      PUT THEM BACK                             IFM2030  204
00A1  E049            LDR   R,9,4       SET DATA                                  IFM2040  205
00A2  E06A            LDR   R,10,6                                                IFM2050  206
00A3  E02B            LDR   R,11,2                                                IFM2060  207
00A4  E03C            LDR   R,12,3                                                IFM2070  208
00A5  6F88            ADD   IS,8,-8     SET ADDRESS FOR DATA FILE                 IFM2080  209
00A6  8D38            STK   DR,8,4      STORE DATA                                IFM2090  210
00A7  C107 0015       BRCL  U,TOP       GO GET MORE PULSES                        IFM2100  211
                      :                                                           IFM2110  212
                                                                                  IFM2120  213
```

```
LOC   OBJECT CODE     CARD IMAGE                                                    CARDNUM
                      .                                                             IFM2130   214
00A9  6011            SPR     AOD   IS,1,1          INCREMENT POINTER               IFM2140   215
00AA  5188                    LDR   RX,11,8,1       FETCH LAST TRAIN START TIME     IFM2150   216
00AB  6011                    ADD   IS,1,1                                          IFM2160   217
00AC  581C                    LDR   RX,12,1,8                                       IFM2170   218
00AD  E026                    LDR   R,6,2           MOVE NEW TOA TO WORKING REGS    IFM2180   219
00AE  E037                    LDR   R,7,3                                           IFM2190   220
00AF  AAB6                    DSUB  R,6,11          CALCULATE SPR                   IFM2200   221
00B0  4041                    LDR   IS,1,4          STORE NEW TOA AND SPR           IFM2210   222
00B1  1812                    STR   RX,2,1,8                                        IFM2220   223
00B2  6011                    AOD   IS,1,1                                          IFM2230   224
00B3  1813                    STR   RX,3,1,8                                        IFM2240   225
00B4  6011                    ADD   IS,1,1                                          IFM2250   226
00B5  1816                    STR   RX,6,1,8                                        IFM2260   227
00B6  6011                    ADD   IS,1,1                                          IFM2270   228
00B7  1817                    STR   RX,7,1,8                                        IFM2280   229
00B8  0847                    BRCS  U,STORE         GO FINISH UPDATING PULSE INFO   IFM2290   230
                      .                                                             IFM2300   231
                      .                                                             IFM2310   232
                      .                                                             IFM2320   233
00B9  601A            INCR    AOD   IS,10,1         ADD 1 TO FILE PW                IFM2330   234
00BA  0827                    BRCS  U,STORE         GO BACK                         IFM2340   235
                      .                                                             IFM2350   236
00BB  6FFA            DECR    AOD   IS,10,-1        SUB 1 FROM FILE PW              IFM2360   237
00BC  C107 003E               BRCL  U,STORE         GO BACK                         IFM2370   238
                      .                                                             IFM2380   239
                      .                                                             IFM2390   240
                      .                                                             IFM2400   241
00BE  6019            INCP    ADD   IS,9,1          ADD 1 TO FILE PRI               IFM2410   242
00BF  0827                    BRCS  U,STACK         GO BACK                         IFM2420   243
                      .                                                             IFM2430   244
                      .                                                             IFM2440   245
00C0  6FF9            DECP    ADD   IS,9,-1         SUB 1 FROM FILE PRI             IFM2450   246
00C1  C107 0043       STEP9   BRCL  U,STACK         GO BACK                         IFM2460   247
                      .                                                             IFM2470   248
                      .                                                             IFM2480   249
                      .                                                             IFM2490   250
                      .                                                             IFM2500   251
                      .       SET HISTOGRAM LIMITS AND DEFINE STORAGE LOCATIONS      IFM2510   252
                      .                                                             IFM2520   253
                      .                                                             IFM2530   254
00C3  0015            HADRS   DC    TOP             IFM2540   255
00C4  00C1                    DC    STEP9           IFM2550   256
                              ORG   RAM             IFM2560   257
1000                  TRACK   DS    2048            IFM2570   258
1800                  EFILE   DS    256             IFM2580   259
1900                  OFLO    DS    128             IFM2590   260
1980  1900            OFLOW   DC    OFLO            IFM2600   261
1981                  TFILE   DS   128              IFM2610   262
1AC1                  NEXTF   DS    1               IFM2620   263
1AC2  1981            PFILE   DC    TFILE           IFM2630   264
1AC3  1AC0                    DC   TFILE+127        IFM2640   265
1AC4                  DATA    DS    1024            IFM2650   266
```

```
LOC   OBJECT CODE      CARD IMAGE                                  CARDNUM
1E04  1A04             DATAR    DC    DATA                        IFM2660   267
1E05  2A03                      DC    DATA+4095                   IFM2670   268
                       FINIS    END                               IFM2680   269
```

| PC | INSTRUCTION | OPER | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | CCR | IMR\|ITR | TOT TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | | | |
| 0010 | SIM S,00CF | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 0.0 |
| 0012 | SIM 1,1A02 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 0.0 |
| 0014 | SIM 1,1E04 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 0.0 |
| 0016 | LDR I,0,1A04 | | 1A04 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 0.9 |
| 0018 | LOR I,15,FFF8 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | FFFB | 4 | 00000000 | 1.8 |
| 001A | LDR I,1,0080 | | . | 0080 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 2.6 |
| 001C | STRM 0,1,1980,1 | 0080 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 4.5 |
| 001E | BRC IS,7,0001 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 5.6 |
| 0021 | ADD IS,0,0004 | | 1A08 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 6.1 |
| 0022 | ADD IS,15,0001 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | FFFC | 4 | 00000000 | 6.5 |
| 0023 | BRC IS,2,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 6.9 |
| 0024 | RET OR,0,4 | 0030 | 1A04 | 0030 | . | . | 0009 | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 11.5 |
| 0025 | LDR R,5,1 | | . | . | . | . | . | 0030 | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 11.9 |
| 0026 | BRC IS,2,FFF9 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 12.4 |
| 0027 | LDR IS,1,0000 | | . | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 12.8 |
| 0028 | STR RX,1,0,1 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 14.8 |
| 0029 | ADD IS,0,0004 | | 1A08 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 15.3 |
| 002A | LOR R,7,5 | | . | . | . | . | . | . | . | 0030 | . | . | . | . | . | . | . | . | 1 | 00000000 | 15.7 |
| 002B | LDR QX,6,1800,7 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 17.6 |
| 0020 | BRC IS,2,0022 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 18.7 |
| 0051 | LOR I,13,FF3F | | . | . | . | . | . | . | . | . | . | . | . | . | . | FF3F | . | . | 4 | 00000000 | 19.6 |
| 0053 | LDR C,6,1A01 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 21.4 |
| 0055 | ADD IS,6,0002 | | . | . | . | . | . | . | 0002 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 21.9 |
| 0056 | AND R,6,13 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 22.3 |
| 0057 | LDR QX,8,1981,6 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 24.2 |
| 0059 | BRC IS,5,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 24.6 |
| 005A | STRM C,6,1A01,1 | 0002 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 26.5 |
| 005C | STR QX,5,1981,6 | 0030 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 28.5 |
| 005E | ADD IS,6,0001 | | . | . | . | . | . | . | 0003 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 28.9 |
| 005F | LDR QX,8,1981,6 | 1040 | . | . | . | . | . | . | . | . | 1040 | . | . | . | . | . | . | . | 1 | 00000000 | 30.8 |
| 0061 | STR QX,6,1800,7 | 0003 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 32.8 |
| 0063 | LDR R,9,4 | | . | . | . | . | . | . | . | . | . | 0009 | . | . | . | . | . | . | 1 | 00000000 | 33.3 |
| 0064 | LDR IS,10,FFF0 | | . | . | . | . | . | . | . | . | . | . | FFF0 | . | . | . | . | . | 4 | 00000000 | 33.7 |
| 0065 | LOR R,11,2 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 34.1 |
| 0066 | LOR R,12,3 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 34.6 |
| 0067 | STR OR,8,4 | 0009 | . | . | . | . | . | . | . | . | 1044 | . | . | . | . | . | . | . | 2 | 00000000 | 39.9 |
| 0068 | BRC IS,7,FF87 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 41.1 |
| 0021 | ADD IS,0,0004 | | 1A0C | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 41.5 |
| 0022 | ADD IS,15,0001 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | FFF0 | 4 | 00000000 | 41.9 |
| 0023 | BRC IS,2,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 42.4 |
| 0024 | RET OR,0,4 | 0029 | 1A08 | 0029 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 46.9 |
| 0025 | LOR R,5,1 | | . | . | . | . | . | 0029 | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 47.4 |
| 0026 | BRC IS,2,FFF9 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 47.8 |
| 0027 | LOR IS,1,0000 | | . | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 48.3 |
| 0028 | STR RX,1,0,1 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 50.3 |
| 0029 | ADD IS,0,0004 | | 1A0C | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 50.7 |
| 002A | LOR R,7,5 | | . | . | . | . | . | . | . | 0029 | . | . | . | . | . | . | . | . | 1 | 00000000 | 51.1 |
| 0028 | LDR QX,6,1800,7 | 0000 | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 53.0 |
| 0020 | BRC IS,2,0022 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 54.1 |
| 0051 | LDR I,13,FF3F | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 55.0 |
| 0053 | LDR C,6,1A01 | 0002 | . | . | . | . | . | . | 0002 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 56.9 |
| 0055 | ADD IS,6,0002 | | . | . | . | . | . | . | 0004 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 57.3 |
| 0056 | AND R,6,13 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 57.8 |
| 0057 | LDR QX,8,1981,6 | 0000 | . | . | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | 2 | 00000000 | 59.6 |
| 0059 | BRC IS,5,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 60.1 |
| 005A | STRM C,6,1A01,1 | 0004 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 61.9 |

| PC | INSTRUCTION | OPER | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | CCR | IMR\|ITR | TOT TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1A0C | 0000 | 0000 | 0000 | 0009 | 0029 | 0004 | 0029 | 0000 | 0009 | FFF0 | 0000 | 0000 | FF3F | 0000 | FFF0 | | | |
| 005C | STR OX,5,1981,6 | 0029 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 63.9 |
| 005E | AOO IS,6,0001 | | . | . | . | . | . | . | 0005 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 64.4 |
| 005F | LDR OX,8,1981,6 | 1080 | . | . | . | . | . | . | . | . | 1080 | . | . | . | . | . | . | . | 1 | 00000000 | 66.3 |
| 0061 | STR OX,6,1800,7 | 0005 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 68.3 |
| 0063 | LOR R,9,4 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 68.7 |
| 0064 | LDR IS,10,FFF0 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 69.1 |
| 0065 | LOR R,11,2 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 69.6 |
| 0066 | LDR R,12,3 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 70.0 |
| 0067 | STK OR,8,4 | 0009 | . | . | . | . | . | . | . | . | 1084 | . | . | . | . | . | . | . | 2 | 00000000 | 75.4 |
| 0068 | BRC IS,7,FF87 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 76.5 |
| 0021 | AOO IS,0,0004 | | 1A10 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 76.9 |
| 0022 | ADO IS,15,0001 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | FFFE | 4 | 00000000 | 77.4 |
| 0023 | BRC IS,2,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 77.8 |
| 0024 | RET OR,0,4 | 0045 | 1A0C | 0045 | 10E1 | . | 0007 | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 82.4 |
| 0025 | LOR R,5,1 | | . | . | . | . | . | 0045 | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 82.8 |
| 0026 | BRC IS,2,FFF9 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 83.3 |
| 0027 | LDR IS,1,0000 | | . | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 83.7 |
| 0028 | STR RX,1,0,1 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 85.7 |
| 0029 | ADO IS,0,0004 | | 1A10 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 86.1 |
| 002A | LOR R,7,5 | | . | . | . | . | . | . | . | 0045 | . | . | . | . | . | . | . | . | 1 | 00000000 | 86.6 |
| 002B | LDR OX,6,1800,7 | 0000 | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 88.4 |
| 002D | BRC IS,2,0022 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 89.6 |
| 0051 | LDR I,13,FF3F | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 90.4 |
| 0053 | LDR O,6,1A01 | 0004 | . | . | . | . | . | . | 0004 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 92.3 |
| 0055 | ADO IS,6,0002 | | . | . | . | . | . | . | 0006 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 92.8 |
| 0056 | AND R,6,13 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 93.2 |
| 0057 | LDR OX,8,1981,6 | 0000 | . | . | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | 2 | 00000000 | 95.1 |
| 0059 | BRC IS,5,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 95.5 |
| 005A | STRM O,6,1A01,1 | 0006 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 97.4 |
| 005C | STR OX,5,1981,6 | 0045 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 99.4 |
| 005E | AOO IS,6,0001 | | . | . | . | . | . | . | 0007 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 99.8 |
| 005F | LOR OX,8,1981,6 | 10C0 | . | . | . | . | . | . | . | . | 10C0 | . | . | . | . | . | . | . | 1 | 00000000 | 101.7 |
| 0061 | STR OX,6,1800,7 | 0007 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 103.7 |
| 0063 | LOR R,9,4 | | . | . | . | . | . | . | . | . | . | 0007 | . | . | . | . | . | . | 1 | 00000000 | 104.1 |
| 0064 | LDR IS,10,FFF0 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 104.6 |
| 0065 | LDR R,11,2 | | . | . | . | . | . | . | . | . | . | . | . | 10E1 | . | . | . | . | 1 | 00000000 | 105.0 |
| 0066 | LDR R,12,3 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 105.4 |
| 0067 | STK OR,8,4 | 0007 | . | . | . | . | . | . | . | . | 10C4 | . | . | . | . | . | . | . | 2 | 00000000 | 110.8 |
| 0068 | BRC IS,7,FF87 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 111.9 |
| 0021 | AOO IS,0,0004 | | 1A14 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 112.4 |
| 0022 | ADO IS,15,0001 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | FFFF | 4 | 00000000 | 112.8 |
| 0023 | BRC IS,2,FFFA | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 113.3 |
| 0024 | RET OR,0,4 | 0010 | 1A10 | 0010 | 24E9 | . | 0028 | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 117.8 |
| 0025 | LDR R,5,1 | | . | . | . | . | . | 001D | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 118.3 |
| 0026 | BRC IS,2,FFF9 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 118.7 |
| 0027 | LDR IS,1,0000 | | . | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 119.1 |
| 0028 | STR RX,1,0,1 | 0000 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 121.1 |
| 0029 | AOO IS,0,0004 | | 1A14 | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 121.6 |
| 002A | LOR R,7,5 | | . | . | . | . | . | . | . | 0010 | . | . | . | . | . | . | . | . | 1 | 00000000 | 122.0 |
| 002B | LDR OX,6,1800,7 | 0000 | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 123.9 |
| 002D | BRC IS,2,0022 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | 125.0 |
| 0051 | LDR I,13,FF3F | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 4 | 00000000 | 125.9 |
| 0053 | LDR O,6,1A01 | 0006 | . | . | . | . | . | . | 0006 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 127.8 |
| 0055 | ADO IS,6,0002 | | . | . | . | . | . | . | 0008 | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 128.2 |
| 0056 | AND R,6,13 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 1 | 00000000 | 128.6 |
| 0057 | LDR OX,8,1981,6 | 0000 | . | . | . | . | . | . | . | . | 0000 | . | . | . | . | . | . | . | 2 | 00000000 | 130.5 |

ATAC INSTRUCTION LEVEL SIMULATION DATE 06/16/74 TIME 17:11:01 PAGE 1

| PC | INSTRUCTION | OPER | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | CCR | IMR | ITR | TOT TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 2A04 | 0003 | 34DE | 0000 | 0000 | 0035 | FFFF | 0000 | 1244 | 0004 | 0013 | 340E | 0000 | 0000 | 0013 | 0000 | | | | |
| 0021 | SIM 6,0000 | | 2A04 | 0003 | 340E | 0000 | 0000 | 0035 | FFFF | 0000 | 1244 | 0004 | 0013 | 340E | 0000 | 0000 | 0013 | 0000 | 2 | 00000000 | | 46246.9 |
| 0023 | SIM 7,0000 | | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | . | 2 | 00000000 | | 46246.9 |

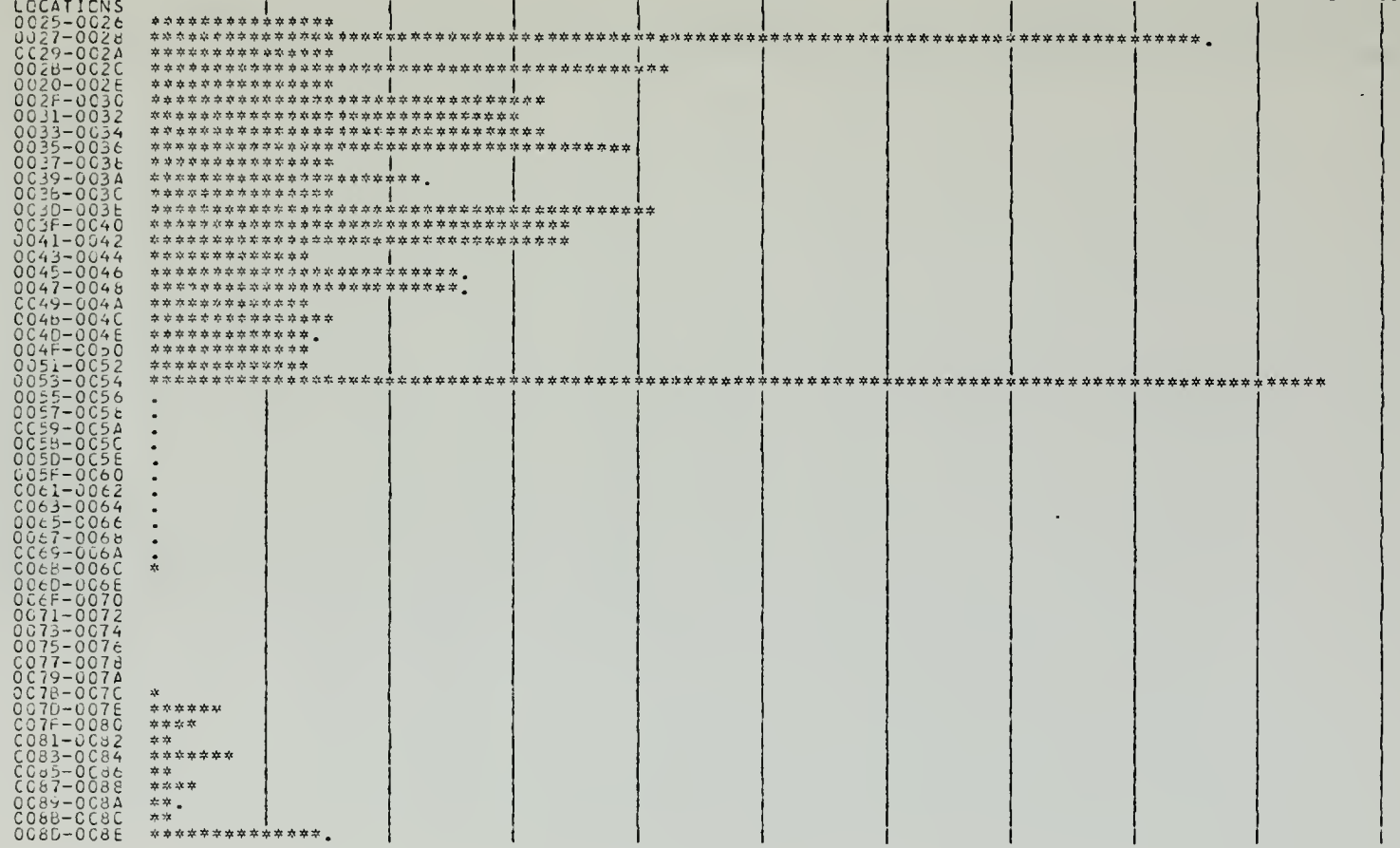
***** NORMAL TERMINATION AT 0023

SIMULATION HISTOGRAM LOCATION VS EXECUTION TIME

DATE 06/16/74 TIME 17:12:44 PAGE 1

SECONDS

* = 6.00E-05 . = 3.00E-05



SIMULATION HISTOGRAM LOCATION VS EXECUTION TIME DATE 06/16/74 TIME 17:12:44 PAGE 2

SECONDS * = 6.00E-05 . = 3.00E-05

```
LOCATIONS          6.0E-04   1.2E-03   1.8E-03   2.4E-03   3.0E-03   3.6E-03   4.2E-03   4.8E-03   5.4E-03   6.0E-03
                   |         |         |         |         |         |         |         |         |         |
0088-0090  .       |         |         |         |         |         |         |         |         |         |
0091-0092  *       |         |         |         |         |         |         |         |         |         |
0093-0094  .       |         |         |         |         |         |         |         |         |         |
0095-0096  .       |         |         |         |         |         |         |         |         |         |
0097-0098  *       |         |         |         |         |         |         |         |         |         |
0099-009A  ****    |         |         |         |         |         |         |         |         |         |
009B-009C  **      |         |         |         |         |         |         |         |         |         |
009D-009E  *******  |         |         |         |         |         |         |         |         |         |
009F-00A0  **      |         |         |         |         |         |         |         |         |         |
00A1-00A2  *       |         |         |         |         |         |         |         |         |         |
00A3-00A4  **      |         |         |         |         |         |         |         |         |         |
00A5-00A6  *       |         |         |         |         |         |         |         |         |         |
00A7-00A8  .       |         |         |         |         |         |         |         |         |         |
00A9-00AA  *       |         |         |         |         |         |         |         |         |         |
00AB-00AC  *       |         |         |         |         |         |         |         |         |         |
00AD-00AE  .       |         |         |         |         |         |         |         |         |         |
00AF-00B0  *       |         |         |         |         |         |         |         |         |         |
00B1-00B2  .       |         |         |         |         |         |         |         |         |         |
00B3-00B4  .       |         |         |         |         |         |         |         |         |         |
00B5-00B6  *       |         |         |         |         |         |         |         |         |         |
00B7-00B8  .       |         |         |         |         |         |         |         |         |         |
00B9-00BA          |         |         |         |         |         |         |         |         |         |
00BB-00BC          |         |         |         |         |         |         |         |         |         |
00BD-00BE          |         |         |         |         |         |         |         |         |         |
00BF-00C0          |         |         |         |         |         |         |         |         |         |
00C1-00C2          |         |         |         |         |         |         |         |         |         |
00C3-00C4          |         |         |         |         |         |         |         |         |         |
00C5-00C6          |         |         |         |         |         |         |         |         |         |
00C7-00C8          |         |         |         |         |         |         |         |         |         |
00C9-00CA  *****   |         |         |         |         |         |         |         |         |         |
00CB-00CC  ****    |         |         |         |         |         |         |         |         |         |
00CD-00CE  **.     |         |         |         |         |         |         |         |         |         |
00CF-00D0          |         |         |         |         |         |         |         |         |         |
00D1-00D2          |         |         |         |         |         |         |         |         |         |
```

DUMP OF PSEUDO-MEMORY SEGMENT RAM FROM 1000 TO 2FFF

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| | LINES 1010 THROUGH 1030 SAME AS ABOVE. | | | | | | | | | | | | | | | |
| 1040 | 0031 | 0030 | 3404 | C000 | 0000 | 0000 | 0000 | 0000 | 0031 | 0031 | 0061 | 0000 | 0031 | 0031 | 0092 | 0000 |
| 1050 | 0031 | 0031 | 00C3 | 0000 | 0031 | 0030 | 00F3 | 0000 | 0031 | 0031 | 0124 | 0000 | 0031 | 0031 | 0155 | 0000 |
| 1060 | 0031 | 0031 | 0186 | C000 | 0031 | 0031 | 0187 | 0000 | 0031 | 0031 | C1E8 | 0000 | 0031 | 0030 | 0218 | 0000 |
| 1070 | 0031 | 0031 | 0249 | 0000 | 0031 | 0031 | 027A | 0000 | 0031 | 00F4 | 036E | 0000 | 0031 | 0031 | 039F | 0000 |
| 1080 | 0017 | 00C6 | 3459 | C000 | 0000 | 0000 | 0000 | 0000 | 0017 | 00C5 | 024F | 0000 | 0017 | 00C5 | 0314 | 0000 |
| 1090 | 0017 | 00C5 | 03D9 | 0000 | 0017 | 00C5 | 049E | 0000 | 0017 | 018A | C628 | 0000 | 0017 | 00C5 | 06E0 | 0000 |
| 10A0 | 0017 | 018A | 0877 | 0000 | 0017 | 0315 | 088C | 0000 | 0017 | 00C5 | CC51 | 0000 | 0017 | 018A | 0008 | 0000 |
| 10B0 | 0017 | 00C5 | 0EA0 | 0000 | 0017 | 018A | 102A | 0000 | 0017 | 00C5 | 10EF | 0000 | 0017 | 00C5 | 1184 | 0000 |
| 10C0 | 0013 | 0075 | 34A4 | 0000 | 0000 | 0000 | 0000 | 0000 | 0013 | 0075 | C0EA | 0000 | 0013 | 0075 | 015F | 0000 |
| 10D0 | 0013 | 0075 | 01D4 | 0000 | 0013 | 0075 | 0249 | 0000 | 0013 | 0076 | C28F | 0000 | 0013 | 0075 | 0334 | 0000 |
| 10E0 | 0013 | 0075 | 03A9 | 0000 | 0013 | 0075 | 041E | 0000 | 0013 | 0075 | C493 | 0000 | 0013 | 0075 | 0508 | 0000 |
| 10F0 | 0013 | 024A | 0752 | 0000 | 0013 | 0076 | 07C8 | 0000 | 0013 | 0075 | C83D | 0000 | 0013 | 0075 | 0882 | 0000 |
| 1100 | 0009 | FFF0 | 01A0 | C000 | 0009 | 002B | 0108 | 0000 | 0009 | 0028 | C203 | 000C | 0009 | 002B | 022E | 0000 |
| 1110 | 00C9 | 0EF0 | 111E | 0000 | 0009 | 002B | 1149 | 0000 | 0009 | 0026 | 1174 | 0000 | 0009 | 0028 | 119F | 0000 |
| 1120 | 00C9 | 0EF0 | 208F | 0000 | 0009 | 0028 | 20BA | 0000 | 0009 | 0028 | 20E5 | 0000 | 0009 | 0E98 | 2F80 | 0000 |
| 1130 | 00C9 | 002A | 2FAA | 0000 | 0009 | 0028 | 2FD5 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| 1140 | 0028 | 0025 | 34C3 | 0000 | 01E2 | 0000 | 0000 | 0000 | 0028 | 0025 | 022C | 0000 | 0028 | 0025 | 0251 | 0000 |
| 1150 | 0028 | 0025 | 0276 | 0000 | 0028 | 0025 | 0298 | 0000 | 0028 | 0025 | 02C0 | 0000 | 0028 | 0025 | 02E5 | 0000 |
| 1160 | 0028 | 0026 | 0308 | 0000 | 0028 | 01E2 | 04E0 | 0000 | 0028 | 0025 | C512 | 0000 | 0028 | 0025 | 0537 | 0000 |
| 1170 | 0028 | 0025 | 055C | C000 | 0028 | 0025 | 0561 | 0000 | 0028 | 0026 | C5A7 | 0000 | 0028 | 0025 | 05CC | 0000 |
| 1180 | 00C7 | 001E | 34C1 | 0000 | 29C2 | 0000 | 269A | 00C0 | 0007 | 0078 | C418 | 0000 | 0007 | 0077 | 048F | 0000 |
| 1190 | 0007 | 001E | 04A0 | 0000 | 0007 | 005A | 0507 | 0000 | 0007 | 001E | 0525 | 0000 | 0007 | 005A | 057F | 0000 |
| 11A0 | 00C7 | 001E | 0590 | C000 | 0007 | 005A | C5F7 | 0000 | 0007 | 001E | 0615 | 0000 | 0007 | C059 | 066E | 0000 |
| 11B0 | 00C7 | 001E | 068C | 0000 | 0007 | 001E | 06AA | 0000 | 0007 | 003C | 06E6 | 000C | 0007 | 001E | 0704 | 0000 |
| 11C0 | 0031 | 004E | 34C8 | 0000 | 124F | 0000 | 0000 | 0000 | 0031 | 004E | 12E8 | 0000 | 0031 | 004E | 1339 | 0000 |
| 11D0 | 0031 | 004E | 1387 | 0000 | 0031 | 004E | 1305 | 0000 | 0031 | 004F | 1424 | 0000 | 0031 | 004E | 1472 | 0000 |
| 11E0 | 0031 | 004E | 14C0 | 0000 | 0031 | 004E | 150E | 0000 | 0031 | 004E | 155C | 0000 | 0031 | 004E | 15AA | 0000 |
| 11F0 | 0031 | 004E | 15F8 | 0000 | 0031 | 1780 | 2078 | 00C0 | 0031 | 004E | 20C6 | 00C0 | 0031 | 004E | 2E14 | 0000 |
| 12C0 | 00C9 | 0020 | 30D7 | C000 | 2138 | 0000 | 0000 | 0000 | 0009 | 00AC | 2293 | 0000 | 0009 | 00AC | 233F | 0000 |
| 1210 | 0009 | 0028 | 236A | 0000 | 0009 | 0056 | 23C0 | 0000 | 0009 | 0028 | 23EB | 0000 | 0009 | 0028 | 2416 | 0000 |
| 1220 | 00C9 | 0056 | 246C | C000 | 0009 | 0028 | 2497 | 0000 | 00C9 | 0028 | 24C2 | 0000 | 0009 | 0055 | 2517 | 0000 |
| 1230 | 00C9 | 0028 | 2542 | 0C00 | 0009 | 0028 | 2560 | 0000 | 0009 | 0056 | 25C3 | 0000 | 0009 | 002B | 25EE | 0000 |
| 1240 | 0004 | 0013 | 34DE | C000 | 2A10 | 0000 | 0000 | 0000 | 0004 | 0014 | 2A44 | 0000 | 0004 | 0013 | 2A57 | 0000 |
| 1250 | 0004 | 0014 | 2A68 | 0000 | 0004 | 0013 | 2A7E | 0000 | 0004 | 0014 | 2A92 | 0000 | 0004 | 0013 | 2AA5 | 0000 |
| 1260 | 0004 | 0014 | 2AB9 | 0000 | 0004 | 0013 | 2ACC | 0000 | 0004 | 0014 | 2AE0 | 0000 | 0004 | C013 | 2AF3 | 0000 |
| 1270 | 0004 | 0014 | 2807 | 0000 | 0004 | 0014 | 2B1B | 00C0 | 0004 | 0013 | 2B2E | 0000 | 0004 | 0014 | 2642 | 0000 |
| 1280 | 00C0 | 00C0 | 0000 | C000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | C000 | 0000 | 0000 | 0000 | 0000 |
| | LINES 1290 THROUGH 1800 SAME AS ABOVE. | | | | | | | | | | | | | | | |
| 1810 | 0000 | 0000 | 0000 | C000 | 0000 | 0000 | 0000 | 0003 | 0000 | 0000 | C000 | 0000 | 0000 | 000B | 0000 | 0000 |
| 1820 | 00C0 | 00C0 | 0000 | 0000 | 0000 | C000 | 0000 | 0005 | 0009 | 0011 | C000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| 1830 | 00C0 | 0000 | 00CC | C000 | 0000 | 0013 | 0000 | 0000 | 0000 | 0000 | C000 | 0000 | 0000 | 000F | 0000 | 0000 |
| 1840 | 0007 | 0000 | 0000 | C000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | C000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| 1850 | C0C0 | 0000 | 000C | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| | LINES 1860 THROUGH 1970 SAME AS ABOVE. | | | | | | | | | | | | | | | |
| 1980 | 0080 | 0000 | 1000 | 0017 | 1040 | 0027 | 1080 | 0040 | 10C0 | 0028 | 1100 | 0010 | 1140 | C045 | 1180 | 003C |
| 1990 | 11C0 | 0029 | 1200 | 0035 | 1240 | 0000 | 1280 | 0000 | 12C0 | 0000 | 1300 | 0000 | 1340 | 0000 | 1380 | 0000 |
| 19A0 | 13C0 | 0000 | 1400 | 0000 | 1440 | 0000 | 1480 | 0000 | 14C0 | 0000 | 1500 | 0000 | 1540 | 0000 | 1580 | 0000 |
| 19B0 | 15C0 | 0000 | 1600 | 0000 | 1640 | 0000 | 1680 | 0000 | 16C0 | 0000 | 1700 | 0000 | 1740 | 0000 | 1780 | 0000 |
| 19C0 | 17C0 | 0000 | 1800 | 0000 | 1840 | 0000 | 1880 | 0000 | 18C0 | 0000 | 1900 | 0000 | 1940 | 0000 | 1980 | 0000 |
| 19D0 | 19C0 | 0000 | 1AC0 | C000 | 1A40 | 0000 | 1A80 | 0000 | 1AC0 | 0000 | 1800 | 0000 | 1B40 | 0000 | 1B80 | 0000 |
| 19E0 | 18C0 | 0000 | 1C00 | C000 | 1C40 | 0000 | 1C80 | 0000 | 1CC0 | 0000 | 1000 | 0000 | 1D40 | 0000 | 1080 | 0000 |
| 19F0 | 10C0 | 0000 | 1EC0 | 0000 | 1E40 | 0000 | 1E80 | 0000 | 1EC0 | 0000 | 1F00 | 0000 | 1F40 | 0000 | 1F80 | 0000 |
| 1AC0 | 1FC0 | 0012 | 1981 | 1A00 | 0000 | 0000 | 0000 | 0031 | C000 | 0000 | 0000 | 0017 | 0000 | 0000 | 0000 | 0013 |

(This page intentionally blank)

# APPENDIX D

## THE IFM DIGITAL PROCESSOR PROGRAM DETAILED DESCRIPTION

### A. PROGRAMMING

The coding used for the ATAC Assembly Language Program was developed by encoding each block of the algorithm flow chart of Figure 1 separately and then linking the blocks together. These coding blocks are described below.

1. LOAD DATA FROM DATA STORAGE. The data loading operation is critical in that it requires considerable execution time and is required for every pulse. The most efficient load instruction is the RETURN which is designed to return stacked values. This instruction automatically fetches a sequence of words and decrements the address pointer. In using this instruction in this application, prior and post operations to the address pointer are required in order to keep the pointer at the proper location. The sequence required is as follows. The actual instructions are indicated.

```
TOP  ADD IS,0,4   Add 4 to the address pointer in
                  register 0.
     ADD  IS,15,1     Increment Program Control Pulse Counter
     BRCS  Z,DONE      Go stop simulation if enough pulses
                       have been analyzed.
```

RET DR,0,4 Starting at the address in register 0 decrement the address and fetch the contents of the address. Repeat until 4 values have been fetched. The returned values are stored in registers 1-4 in the order they appear in memory. On completion of the operation, the address pointer value is 4 less than at the beginning.

2. DATA AVAILABLE? To test the contents of register 1 an operation has to be performed to set the condition code register. The operation selected is to move the contents to register 5 for future operations. Register 1 will later be used as a general zero contents and indexing register.

LDR R,5,1 Load the contents of register 1 into register 5.

BRCS Z,TOP If the contents of register 5 (frequency) are zero, branch to the beginning (TOP) and access the same data block again. This sets up the no input data idle loop.

3. ZERO DATA LOCATIONS, INCREMENT ADDRESS POINTER. The frequency word location in the data memory is set to zero indicating the data for that pulse has been read. The address

pointer in register 0 is incremented to point at the location of the data words of the next pulse.

```
    LDR  IS,1,0      Load 0 into register 1
    STR  RX,1,0,1    Store the contents of register 1 in the
                     location whose address is the sum of
                     registers 0 and 1.
    ADD  IS,0,4      Add 4 to the contents of register 0.
                     Set the data file pointer for the next
                     pulse.
```

4. FIND CONTACT FILE NUMBER. The frequency word in register 5 is used as a relative address for entry into the frequency file. The frequency word is copied into register 7 so that it can be saved in the event multiple files exist for this frequency.

```
    LDR  R,7,5       Load the contents of register 5 into
                     register 7.
FLINK LDR DX,6,EFILE,7  Load register 6 with the contents of
                     the location whose address is the sum
                     of EFILE, the base address of the fre-
                     quency file, and register 7.
```

5. FILE EXISTS? If no file for this frequency has been established, the contents of the frequency file is 0. A branch on condition zero instruction is used.

```
    BRCS  Z,STAR1    If frequency word is zero, branch to
                     STAR1 where a file will be created.
```

6. FIND CONTACT FILE ENTRY ADDRESS. This instruction is executed only if the frequency file already exists. The

lower byte of the frequency file entry is the relative address of the proper contact file entry. This file has two words, the second word is the pulse train file entry address. The coding to achieve the linking is shown below. At the start of the sequence, the frequency file contents are in register 6.

```
LDLB R,14,6        Load the lower byte of register 6
                   into register 14
LDR DX,8,TFILE,14  Load register 8 with the contents
                   of the address formed by adding the
                   contents of register 14 and the value
                   of TFILE.
```

7. PW = FILE PW? The most efficient coding of this algorithm step depends on the relative frequency that the condition is met. If the average condition is that the incoming pulse width at this point will be equal to the file pulse width, the most efficient coding will return the first two words of the pulse train file together. If the branch is to be taken, the most efficient coding will return just the single pulse width word. The latter condition was chosen. Using the entry address contained in register 8, the first pulse width of the file is returned, compared to the incoming pulse width and the appropriate branch taken. The ATAC instruction set contains a compare between limits instruction which would perform the desired operation but it does not have a mode that is compatible with the program. The use of the ATAC instruction requires

enough manipulation of the contents of registers so that it requires more execution time than directly coding the individual steps. The coding for this operation is as follows.

```
    LDR  RX,9,1,8    Load register 9 with the contents of
                     the address formed by adding register
                     8 and register 1 (register 1 still is 0)
    LDR  R,13,9      Load register 13 with the contents of
                     register 9, This is done to eliminate
                     the requirements to load register 9
                     again.
    SUB  R,13,4      Subtract the contents of register 4
                     (incoming pulse width) from the con-
                     tents of register 13 and place the
                     results in register 13.
    BRCS P,TEST      If the result is positive branch to
                     the positive test (skip the following
                     instruction).
    LAC  R,13,13     Load the arithmetic complement of
                     register 13 into register 13.
TEST CMP IS,13,4     Compare the magnitude of the pulse
                     width difference in register 13 with
                     the threshhold value 4.
    BRCS P, NTRY     If the contents of register 13 is
                     greater than 4, the branch is taken
                     to location NTRY which is the address
                     of the beginning of the LOOK FOR NEXT
                     FILE routine.
```

8. HAS PRI BEEN CALCULATED? The contents of the location following the pulse width word in the pulse train file is the PRI control word. If its contents are positive, a PRI has been calculated for the pulse train and the contents of the word is the value of the PRI. If the contents of the word is 0, the pulse train has an unstable PRI. In both of these cases, the program should proceed to update the pulse train file data. If the contents of the PRI word are less than zero, the pulse train file is still building and the program should proceed to the build file routine.

```
ADD  IS,1,1      Add 1 to the contents of register 1.
                 Register 1 will be used an an index
                 to step to the next word in the pulse
                 train file.
LDR  RX,10,1,8   Load register 10 with the contents of
                 the address formed by adding the con-
                 tents of registers 1 and 8.
BRCS N,BUILD     Branch to the beginning of the build
                 file routine if the contents of
                 register 10 are negative.
```

9. TIME SINCE LAST PULSE LARGE? The UPDATE routine now calculates data to be used to update the pulse train file. This first operation requires the return of the next two words of the pulse train file which contain the time that the last update occurred. The difference between the last update time and the current pulse time of arrival contained in registers 3 and 4 is used to determine whether the

PRI will be updated or SPR calculated. The test used is whether the upper half of the double precision difference of the two times of arrival is zero. This results in the algorithm considering all time intervals greater than 0.8 seconds as SPR and all intervals less than 0.8 seconds as PRI.

```
UPDATE  ADD   IS,1,1    Step the indexing register to fetch
                        the next word
        LDR RX,11,1,8   Load the lower half of the time word
                        into register 11
        ADD   IS,1,1    Step the index register again
        LDR   RX,12,1,8 Load the upper half of the time word
                        into register 12
        LDR   R,6,2     Load the contents of register 2 into
                        register 6
        LDR   R,7,3     Load the contents of register 3 into
                        register 7.  These two instructions
                        copy the double precision new time
                        into registers 6 and 7 so that the
                        actual time value is retained in
                        registers 2 and 3
        DSUB  R,6,11    Double precision subtract the contents
                        of registers 11 and 12 from the contents
                        of registers 6 and 7 and place the
                        results in registers 6 and 7.
```

```
BRCS N,TOP      If the result is negative, disregard
                this pulse by going to the beginning
                to get the next pulse
BRCS NZ,SPR     If the upper half of the double
                precision difference is not zero,
                branch to the SPR calculation routine.
LDUB  R,13,6    Load the upper byte of register 6 into
                register 13.
CMP   IS,13,16  Compare the contents of register 13
                with the magnitude 16.  If the con-
                tents of register 13 is greater than
                16, the time interval between pulses
                is greater than 52.4 milliseconds.
BRCS P,SPR      If the time interval is greater than
                52.6 milliseconds, go calculate SPR.
```

10. UPDATE PRI,PW AND TIME. PRI and PW are updated by determining the relative size of the new value and the old. If the new value is greater than the old, 1 is added to the old and vice versa. If there is no difference, no change is made. The time is simply loaded into registers 11 and 12. On completion of this step, the four updated pulse train file words are contained in registers 9 through 12.

```
SUB  R,6,10     Subtract the contents of register 10
                from the contents of register 6 and
                place the result in register 6.
```

```
        BRCS P,INCR     If the difference is greater than
                        zero, the new time difference is
                        larger.  Branch to the INCRease routine
                        which will add one to the old PRI.
        BRCS N,DECR     If the difference is less than zero,
                        the new time difference is smaller.
                        Branch to the DECRease routine which
                        will subtract one from the old PRI.
STORE   LDR  R,11,2     Load the new time located in register
                        2 into register 11
        LDR  R,12,3     Load the upper half of the new time
                        located in register 3 into register 12.
        SUB  R,4,9      Subtract the contents of register 9
                        from the contents of register 4 and
                        place the results in register 4.  This
                        subtracts the old pulse width from the
                        new pulse width
        BRCS P,INCP     If the result is positive, the new
                        pulse width is greater.  Branch to the
                        increase pulse width routine (INCP)
                        which will add 1 to the old pulse
                        width.
        BRCS N,DECP     If the result is negative, the new
                        pulse width is smaller.  Branch to the
                        decrease pulse width routine (DECP)
                        which will subtract 1 from the old
                        pulse width.
```

11. STORE UPDATED DATA. The four updated file words are placed in memory using a stack instruction. After the stack operation, the program branches unconditionally to the beginning to get the next pulse.

```
STACK  STK DR,8,4     Store the next four registers (9-12)
                      into memory locations beginning with
                      the address contained in register 8.
                      Add 4 to the contents of register 8
                      on completion.
       BRCS U,TOP     Unconditionally branch to the beginning
                      (TOP) to get the next pulse.
```

12. START FILE. The start file routine (STAR1) is entered when the frequency file word is zero indicating that no pulse train file exists for the frequency detected. It is also entered after linking information has been prepared when the first file was not the correct one. The routine searches for the next empty pulse train file and then enters linking information into the frequency and contact files. The data is then loaded into the first entry location with -16 stored in the PRI word indicating one pulse for this file has been received. The location NEXTF is used to point to the last file activated so that the files are activated in sequence. A mask is used so that the values in NEXTF will always be modulus 64. Files are cleared by the main classification computer.

```
STAR1  LDR I,13,-193  Load modulo 64 mask into register 13
```

```
    LDR  D,6,NEXTF     Load the contents of location NEXTF
                       into register 6
ANOT ADD  IS,6,2       Register 6 contains the last file
                       used.  Add 2 to get the address of
                       the next file entry.
    AND R,6,13         Form the logical product of the contents
                       of registers 6 and 13 and place the
                       results in register 6.  The contents
                       of register 6 are now modulo 64.
    LDR  DX,8,TFILE,6  Load in register 8 the contents of
                       the location whose address is the sum
                       of TFILE and register 6.  This is the
                       address of the frequency associated
                       with the next pulse train file.
    BRCS NZ,ANOT       If the frequency value is not zero,
                       branch to location ANOT which is the
                       instruction to increment NEXTF.
    STR  D,6,NEXTF     Store the contents of register 6 in
                       location NEXTF.  This stores the
                       pointer value for the pulse train file
                       which is now being activated.
    STR DX,5,TFILE,6   Store the contents of register 5 in
                       the location whose address is the sum
                       of TFILE and register 6.  This speci-
                       fies the frequency of the pulse train
                       file and also indicates that the file
                       is busy.
```

```
ADD  IS,6,1          Add one to the contents of register 6.
LDR  DX,8,TFILE,6    Load register 8 with the contents of
                     the location whose address is the sum
                     of TFILE and register 6.  This is
                     the second word of the contact file
                     entry and is the actual address of the
                     first word of the pulse train file.
STR  DX,6,EFILE,7    Store the contents of register 6 in
                     the location whose address is the sum
                     of EFILE and register 7.  This sets the
                     link between the frequency word and
                     the pulse train file entry address.
LDR  R,9,4           Load register 9 with the contents of
                     register 4 (pulse width).
LDR  IS,10,-16       Load register 10 with the number -16
                     (pulse counter)
LDR  R,11,2          Load register 11 with the contents of
                     register 2 (time, lower half)
LDR  R,12,3          Load register 12 with the contents of
                     register 3 (time upper half).  The
                     pulse data is contained in registers
                     9 through 12.
STK  DR,8,4          Store the contents of registers 9
                     through 12 in memory locations beginning
                     with the address contained in register 8.
BRCS  U,TOP          Branch unconditionally to the beginning
                     (TOP) to get the next pulse.
```

13. WRONG FILE. The routine NTRY is entered when the pulse width of the previous pulse train file was not within limits. This routine establishes linking to the next pulse train file by taking the upper half of the frequency file word and placing it into register 7 and branching to FLINK if a file exists or branching to routine STAR2 if another file has not been started. FLINK is the execution step that enters the frequency file. The contents of register 7 will now be the overflow location in the file. STAR2 establishes the initial linking for establishing multiple files for the same frequency.

```
NTRY LDUB R,14,6      Load register 14 with the upper byte
                      of the contents of register 6.
     BRCS Z,STAR2     If the upper byte of register 6 was
                      zero, another pulse train file has
                      not been established for this frequency.
                      Branch to STAR 2 and establish one.
     LDR  R,7,14      Load register 7 with the contents of
                      register 14.
     BRCS U.FLINK     Unconditionally branch to FLINK.  FLINK
                      is the location of the FIND CONTACT
                      FILE NR sequence.
```

14. START FILE (2) The routine STAR2 is entered when two or more files are established for one frequency. On entry, the lower byte of register 6 always contains the relative address of the last frequency file accessed and the upper byte is always zero indicating that there are no

more files in the chain. STAR2 sets the upper half of this entry word equal to the address of the next entry for this frequency and stores it in the word's proper location. STAR2 then branches to routine STAR1 which finds the next empty pulse train file.

| | |
|---|---|
| LDR D,14,OFLOW | Load register 14 with the contents of location OFLOW. OFLOW contains the relative address of the next available frequency file overflow location. |
| XCHB R,13,14 | Load register 13 with the contents of register 14 with the bytes exchanged. This places the overflow location relative address in the upper byte of register 14. |
| ADD R,6,13 | Add the contents of register 13 to the contents of register 6 and place the results in register 6. This assembles a word in register 6 which consists of two separate bytes. The lower byte is the relative address of the frequency file entry that lead to the incorrect pulse train file. The upper byte is the relative address of the frequency file entry which will lead to the correct pulse train file for the pulse width of the current pulse. |

```
STR DX,6,EFILE,7  Store the contents of register 6
                  in the location whose address is the
                  sum of EFILE and register 7.
LDR  R,7,14       Put proper relative address into
                  register 7
ADD  IS,14,1      Add one to the contents of register 14
STR  D,14,OFLOW   Store the contents of register 14 in
                  location OFLOW.  The contents of OFLOW
                  now point to the next available over-
                  flow location.
BRCS U,STAR1      Unconditionally branch to STAR1 where
                  the next pulse train file will be
                  activated.
```

15. ENOUGH PULSES TO CALCULATE PRI? Routine BUILD is entered when it has been determined that the PRI word of a pulse train file is less than zero indicating the file is still building. The number one is added to the negative pulse counter and it is tested. If the counter is now zero, the current pulse is the last one needed to calculate the PRI and the program branches to the calculate PRI routine (PRI). If the counter is still negative, the program then continues to add the pulse data to the pulse train file.

```
ADD  IS,10,1      Add 1 to the contents of register 10.
BRCS Z,PRI        If the counter is now zero, branch
                  to the calculate PRI routine (PRI).
```

16. ADD INFORMATION TO THE DATA SET. The pulse counter is stored. The proper location of the current pulse data

is determined by adding 16 to the pulse counter value. The result is the number of the pulse in the current pulse train. This number is then multiplied by 4 (shifted left 2 bits) to determine the address of the first data word relative to the start of the file. The data is then stored after checking to ensure it is valid data.

| | |
|---|---|
| STR RX,10,8,1 | Store the contents of register 10 in the location whose address is the sum of registers 8 and 1. Register 8 contains the address of the first location of the pulse train file. Register 1 contains the number 1. The pulse counter is stored in the second word of the file. |
| ADD IS,10,16 | Add 16 to the contents of register 10. The result is the number of the slot for the current pulse data. |
| SHS LL,10,2 | Shift the contents of register 10 left two bits. This multiplies the contents of register 10 by 4. |
| ADD R,8,10 | Add the contents of register 10 to the contents of register 8 and place the results in register 8. This forms the actual address of the first word of the locations for the current pulse. |
| LDR R,5,8 | Load register 5 with the contents of register 8. |

| | |
|---|---|
| ADD IS,5,-2 | Subtract 2 from the contents of register 5. This forms the address of the time of arrival of the previous pulse of the current pulse train. |
| LDRM DR,5,2 | Load registers 6 and 7 with the contents of two adjacent locations whose addresses start at the contents of register 5. |
| LDR R,11,2 | Load register 11 with the contents of register 2. |
| LDR R,12,3 | Load register 12 with the contents of register 3. The time of arrival of the current pulse has been copied into registers 11 and 12. |
| DSUB R,2,6 | Double precision subtract the contents of registers 6 and 7 from the contents of registers 2 and 3 and place the results in registers 2 and 3. This puts the time interval between pulses in registers 2 and 3. |
| BRCS N,TOP | If the time difference is negative, branch to the beginning disregarding the current pulse. |
| BRCS Z,GOOD | If the upper half of the time difference is zero, the time is considered a PRI and the next instruction is not executed. |

```
    LDR  IS,2,0      Load register 2 with the number 0.
                     This will cause zero to be stored for
                     the interval between pulses.
GOOD  LDR  R,10,2    Load the single precision value of
                     the interval between pulses into
                     register 10 from register 2.
    LDR  R,9,4       Load register 9 with the contents of
                     register 4 (pulse width). The contents
                     of registers 9-12 is now the data
                     associated with the current pulse.
    STK  DR,8,4      Store the contents of registers 9-12
                     in locations whose addresses start with
                     the contents of register 8. Add 4 to
                     the contents of register 8.
    BRCS U,TOP       Unconditionally branch to the beginning
                     (TOP) to get the next pulse.
```

17. CALCULATE AVERAGE PRI AND PW. When the seventeenth pulse of a pulse train file is received the program enters PRI which calculates the average PRI and pulse width of the pulse train file. The time interval between the last two pulses is calculated and then all sixteen intervals and the last sixteen pulse widths are added. The program then branches to PEND where the sums are divided by 16 to form the average.

```
PRI  LDR  IS,1,-15   Load register 1 with the number 15.
                     This is used as a counter
```

```
    ADD  IS,8,64     Add 64 to the contents of register 8.
                     Register 8 now contains the required
                     address for the RETURN instruction to
                     return the last pulse train file data
                     set.
    RET  DR,8,4      Load registers 9-12 with the last data
                     set.
    LDR  R,6,2       Load register 6 with the contents of
                     register 2.
    LDR  R,7,3       Load register 7 with the contents of
                     register 3.  The time of arrival of
                     the current pulse is copied into regis-
                     ters 6 and 7.
    DSUB R,6,11      Double precision subtract the contents
                     of registers 11 and 12 from the contents
                     of registers 6 and 7.  The last time
                     interval is contained in registers 6
                     and 7.
    BRCS N,TOP       If the time difference is negative,
                     disregard this pulse.
    LDR  IS,11,0     Load 0 into register 11.
    LDR  R,12,6      Set PRI standard.
LP2 ADD  R,4,9       Add the contents of register 9 to the
                     contents of register 4 and place the
                     sum in register 4.  This accumulates
                     the sum of the pulse widths.
```

```
    DADD  R,6,10      Double precision add the contents of
                      registers 10 and 11 to the contents
                      of registers 6 and 7.
    ADD   IS, 1,1     Add 1 to the contents of register 1.
                      This steps the counter.
    BRCS  Z,PEND      If the counter is zero, all pulse data
                      has been summed.  Branch to the divide
                      routine (PEND).
    ADD   IS,8,-2     Set address for the next data set.
    RET   DR,8,2      Return the pulse width and the time
                      interval from the previous data set.
    SUB   R,10,12     Find the difference between this time
                      interval and the PRI standard.
    BRCS   P,LP3      If the difference is positive, test
                      the results at LP3.
    CMP   IS,10,-4    Compare the difference, when negative,
                      with the magnitude -4.
    BRCS   N,NG       If the difference is less than -4,
                      go throw the pulse out.
    ADD   R,10,12     Restore the time interval value in
                      register 10.
    BRCS   U,LP2      Unconditionally branch to LP2 to sum
                      the data.
LP3 CMP   IS,10,4     If the difference between the time
                      interval and the PRI standard is posi-
                      tive, compare the difference to the
                      magnitude 4.
```

```
    BRCS  P,NG       If the difference is greater than 4,
                     go throw the pulse out.
    ADD   R,10,12    Restore the time interval value in
                     register 10.
    BRCS  U,LP2      Unconditionally branch to LP2 to sum
                     the data.
NG  LDR   R,10,12    Replace the time interval with the
                     value of the PRI standard.
    BRCS  U,LP2      Unconditionally branch to LP2 and sum
                     the data.
PEND SHD  RA,6,4     Double precision shift right regis-
                     ters 6 and 7 4 bits (divide  by 16).
    SHS   RA,4,4     Shift register 4 right 4 bits to
                     divide by 16.
    RET   DR,8,2     Return the first time of arrival in
                     the file.
    ADD   IS,8,2     Set the address for the second data
                     block.
    LDR   IS,11,0    Load 0 into register 11.
    LDR   IS,12,0    Load 0 into register 12.
    STK,DR,8,4       Store the contents of registers 9-12
                     in locations whose addresses start
                     with the contents of register 8.  This
                     stores the times required to calculate
                     SPR and sets the initial value of
                     SPR to 0.
```

```
    LDR  R,9,4       Load register 9 with the contents of
                     register 4.
    LDR  R,10,6      Load register 10 with the contents of
                     register 6.
    LDR  R,11,2      Load register 11 with the contents of
                     register 2.
    LDR  R,12,3      Load register 12 with the contents of
                     register 3.  The entire data set is
                     loaded into registers 9-12.
    ADD  IS,8,-8     Set the address of the first data block
                     of this pulse train.
    STK  DR,8,4      Store the contents of registers 9-12
                     in locations whose addresses start with
                     the contents of register 8.  Add 4 to
                     the contents of register 8.
    BRCL U,TOP       Branch to the beginning to get the next
                     pulse.
```

18. CALCULATE SPR AND UPDATE TIME. Routine SPR is entered when a large time difference is noted between the current pulse and the previous pulse of a closed pulse train file. The time that the last sequential group of pulses arrived is accessed and subtracted from the current time to determine the antenna scan time. The current time and the SPR are stored in the file.

```
SPR ADD  IS,1,1      Increment address pointer.
    LDR  RX,11,8,1   Load register 11 with the contents of
                     the location whose address is the sum
```

```
                    of register 8 and register 1.  This is
                    the start of the time that the last
                    antenna scan was received.
ADD   IS,1,1        Load second half of this time word
LDR   RX,12,8,1     into register 12.
LDR   R,6,2         Move the new time of arrival into
LDR   R,7,3         working registers.
DSUB  R,6,11        Double precision subtract the contents
                    of registers 11 and 12 from the contents
                    of registers 6 and 7.  This is the SPR
                    value.
LDR   IS,1,4        Set the address pointer
STR   RX,2,1,8      Store the new time of arrival
ADD   IS,1,1
STR   RX,3,1,8
ADD   IS,1,1
STR   RX,6,1,8      Store the value of the SPR
ADD   IS,1,1
STR   RX,7,1,8
BRCS  U,STORE       Unconditionally branch to STORE and
                    finish updating the pulse information.
```

## APPENDIX E

## DATA SIMULATION PROGRAM GLOSSARY OF VARIABLES AND FORTRAN LISTING

### DATA SIMULATION PROGRAM GLOSSARY OF VARIABLES

| | |
|---|---|
| ANG(I,J) | True pointing angle of the antenna on the Jth emitter on the Ith platform |
| ANT(I,J0 | Antenna scan rate of the Jth emitter on the Ith platform |
| BIN | IFM frequency bin size |
| BEAM | Input value of antenna beam width |
| BNG(I) | True bearing of the Ith platform |
| BNGI | Computation value of BNG(I) |
| BW(I,J) | Antenna beam attenuation factor of the Jth emitter on the Ith platform |
| CON1 | Conversion factor used in converting from antenna beam width in degrees to the antenna attenuation factor |
| CON2 | Conversion factor used in converting from transmitter peak power to transmitter total effective radiated power |
| CSEI | Input value of the course of the Ith platform |
| CSEO | Input value of the course of the IFM |
| D1 | Distance to CPA |
| D2 | Distance to the maximum range at which an emitter can be detected |

| | |
|---|---|
| DEGR | Conversion factor to convert angles in degrees to angles in radians |
| DEL | Time after the start of the simulation that an emitters first pulse is transmitted or that an emitter is within its maximum detection range |
| F(I,J) | Transmitted frequency of the Jth emitter on the Ith platform |
| FMAX | Maximum receive frequency of the IFM |
| FMIN | Minimum receive frequency of the IFM |
| FREQ | Input value of the operating frequency of an emitter |
| FSCALE | Conversion factor for converting from Mega units to units |
| GAIN | Input value of antenna gain |
| ICT | Counter for number of pulses received by the IFM |
| IF(I,J) | Selector for the desired type of frequency variation |
| INEXT(L) | Platform subscript of the second or more pulses when more than one pulse is received simultaneously by the IFM |
| IPRI(I,J) | Selector for the desired type of PRI variation |
| IPT(I,J) | Selector for the desired type of output power variation |

| | |
|---|---|
| IPW(I,J) | Selector for the desired type of pulse width variation |
| ISCAN(I,J) | Selector for the desired type of scan rate variation |
| ITEMP | Platform subscript of the current pulse received by the IFM |
| JNEXT(L) | Emitter subscript used for multiply received pulses |
| JTEMP | Emitter subscript of the current pulse received by the IFM |
| N | Bin number of the frequency of the received pulse |
| NBIN | Number of frequency bins in the bandwidth of the IFM |
| NGAUS | Integer calling argument for the Gaussian Distribution Random Number Generator |
| NMTR(I) | Number of emitters on the Ith platform |
| NOUT(K) | Array used to prepare output data for the ATAC Simulator |
| NPW | Integer representation of pulse width as required by the ATAC Simulator |
| NTIME | Integer representation of simulation time |
| NTIME1 | Lower half of the double precision time value required by the ATAC Simulator |
| NTIME2 | Upper half of the double precision time value required by the ATAC Simulator |

| | |
|---|---|
| NUNI | Integer calling argument for the Uniform Distribution Random Number Generator |
| PI | 3.1415926 |
| PI2 | PI/2 |
| POWER(I,J) | Total effective transmitted power of the Jth emitter on the Ith platform |
| POWIJ | Calculation value of POWER(I,J) |
| PRI(I,J) | PRI of the Jth emitter on the Ith platform |
| PRIN | Input value of PRI |
| PSCALE | Conversion factor for converting from milli units to units |
| PTRANS | Input value of transmitter power |
| PULSE | Calculation value of PW(I,J) |
| PW(I,J) | Pulse width of the Jth emitter on the Ith platform |
| R | Range of the Ith platform |
| RCVBNG | Bearing from which the current pulse arrived |
| RCVR | Frequency of the received pulse |
| RECSIG | Signal level of the received pulse |
| RCSE | Direction of Relative Motion |
| RDEG | Conversion factor for converting angles in radians to angles in degrees |
| RMAX | Maximum range at which an emitter can be detected |
| SCAN | Subroutine that calcualtes an antenna true pointing angle and its gain in the direction of the IFM |

| | |
|---|---|
| SCNMIN | The minimum value of antenna gain for which an emitter can be detected by the IFM |
| SEND(I,J) | Time of transmission of a pulse from the Jth emitter on the Ith platform |
| SIGFRQ(I,J) | Frequency of the transmitted pulse |
| SPDI | Input speed of the Ith platform |
| SPDO | Input speed of the IFM |
| TF | Simulation stop time |
| THETAI | Bearing of the CPA of the Ith platform |
| THRESH | Threshhold sensitivity of the IFM |
| TINT | Minimum time measurement capability of the system utilizing the IFM |
| TNEXT(I,J) | Time of arrival at the IFM of the next pulse transmitted by the Jth emitter on the Ith platform . |
| TO | Simulation start time |
| TOF | Time required for the propogation of a pulse from an emitter to the IFM |
| TSCALE | Conversion factor for converting from Kilo units to units |
| TSKTS | Conversion factor for converting from knots to yards per second |
| TT | Calculation value of TNEXT(I,J) |
| TWOPI | 2*PI |
| UPDATE | Subroutine that calculates the time of arrival of the next pulse from an emitter |
| VL | Velocity of Light |

| | |
|---|---|
| VTI | Total relative velocity of the Ith platform |
| VX(I) | Relative velocity of the Ith platform in the X direction |
| VXI | Calculation value of VX(I) |
| VXO | Velocity of the IFM in the X direction |
| VY(I) | Relative velocity of the Ith platform in the Y direction |
| VYI | Calculation value of VY(I) |
| VYO | Velocity of the IFM in the Y direction |
| WSCALE | Conversion factor for converting from micro units to units |
| X(I) | Relative X position of the Ith platform |
| XI | Input X position of the Ith platform |
| XO | Input X position of the IFM |
| Y(I) | Relative Y position of the Ith platform |
| YDS | Conversion factor for converting knots to yards per second |
| YI | Input Y position of the Ith platform |
| YO | Input Y position of the IFM |
| Z(I) | Relative Z position of the Ith platform |
| ZI | Input Z position of the Ith platform |
| ZO | Input Z position of the IFM |

DATA SIMULATION PROGRAM DATA DECK SETUP

1. The data deck for use of the DSP is segregated into four basic parts.

    a. (Three cards) IFM Platform Information

    b. (One card) Emitter Platform Information

    c. (Two cards) Emitter Information

    .
    .
    .

    (Up to five emitters)

    .

    b. (One card) Emitter Platform Information

    c. (Two cards) Emitter Information

    .
    .
    .

    (Up to ten platforms)

    d. (One card) End of data card with any negative number in columns 21-30

2. The format for each required card is shown below.

    a. IFM Platform Information

| Card | Columns | Format | Variable | Units |
|---|---|---|---|---|
| 1 | 1-10 | I10 | NUNI | Odd Integer Right Justified |
| | 11-20 | I10 | NGAUS | Odd Integer Right Justified |
| | 21-30 | F10.0 | VL | $10^6$ YDS/SEC |
| | 31-40 | F10.0 | TO | Seconds |
| | 41-50 | F10.0 | TF | Seconds |
| | 51-60 | F10.0 | TINT | Microseconds |
| | 61-70 | F10.0 | THRESH | dbm |

| | | | | |
|---|---|---|---|---|
| 2 | 1-10 | F10.0 | XO | Nautical miles |
| | 11-20 | F10.0 | YO | Nautical miles |
| | 21-30 | F10.0 | ZO | Feet |
| | 31-40 | F10.0 | CSEO | Degrees clockwise from North |
| | 41-50 | F10.0 | SPDO | Knots |
| | 51-60 | F10.0 | FMAX | MHZ |
| | 61-70 | F10.0 | FMIN | MHZ |
| 3 | 1-10 | I10 | NBIN | Integer Right Justified |

b. Emitter Platform Information

| Card | Columns | Format | Variable | Units |
|---|---|---|---|---|
| 1 | 1-10 | F10.0 | XI | Nautical miles |
| | 11-20 | F10.0 | YI | Nautical miles |
| | 21-30 | F10.0 | ZI | Feet |
| | 31-40 | F10.0 | CSEI | Degrees clockwise from North |
| | 41-50 | F10.0 | SPDI | Knots |
| | 51-51 | I1 | NMTR(I) | Integer |

c. Emitter Information

| Card | Columns | Format | Variable | Units |
|---|---|---|---|---|
| 1 | 1-10 | F10.0 | FREQ | MHZ |
| | 11-20 | F10.0 | PRI | milliseconds |
| | 21-30 | F10.0 | PW | microseconds |
| | 31-40 | F10.0 | PTRANS | Kilowatts |
| | 41-50 | F.0.0 | BEAM | Degrees |

| | | | | |
|---|---|---|---|---|
| | 51-60 | F10.0 | GAIN | db |
| | 61-70 | F10.0 | SCAN | Seconds per revolution |
| 2 | 1 | I1 | IF(I) | Integer |
| | 2 | I1 | IPRI(I) | Integer |
| | 3 | I1 | IPW(I) | Integer |
| | 4 | I1 | IPT(I) | Integer |
| | 5 | I1 | ISCAN(I) | Integer |

d. End of data card

Columns 21-30 Any negative number

3. The program listed herein does not include coding for use of the parameter variations. This feature can easily be added by using utility subroutines GAUS and RANDU or any other random number generation scheme.

4. The program listed herein contains coding for only two types of antenna scan--circular and steady. Coding for additional types was being prepared when it was determined what the run times for the ATAC Simulations were going to be. As the simulations take very long periods to run on modest data sets, the addition of the additional features was delayed. Coding to add additional scan types can be readily accomplished.

5. The units of the input variables were selected so that values could be coded without the need for using the exponential format. It is felt that this feature will help reduce data deck format errors by allowing the input data

to be placed anywhere within the 10 column field as long as a decimal place is used somewhere.

```
C                                                                       DSP 0000
C                                                                       DSP 0010
C                 DATA SIMULATION PROGRAM (DSP)                         DSP 0020
C                                                                       DSP 0030
C                                                                       DSP 0040
C     THE DSP GENERATES SIMULATION DATA FOR TESTING IFM DIGITAL         DSP 0050
C     PROCESSOR ALGORITHMS.  FOR DETAILED EXPLANATION OF THE PROGRAM    DSP 0060
C     AND INSTRUCTION FOR ITS USE, SEE NPS THESIS:                      DSP 0070
C                                                                       DSP 0080
C                 ELECTRONIC WARFARE APPLICATIONS OF                    DSP 0090
C                 SPECIAL PURPOSE MICROPROGRAMMABLE MINICOMPUTERS       DSP 0100
C                 LCDR  P. D. FRAZER, JUNE 1974                         DSP 0110
C                                                                       DSP 0120
C                                                                       DSP 0130
      COMMON X(10),Y(10),Z(10),VX(10),VY(10),BNG(10),F(10,5),TINT,VXO,VYDSP 0140
     10,THRESH,PI,PI2,SCNMIN,PRIN,SIGFRQ(10,5),PRI(10,5),PW(10,5),POWER(DSP 0150
     210,5),BW(10,5),SEND(10,5),TNEXT(10,5),SIGNAL(10,5),ANT(10,5),ANG(1DSP 0160
     30,5),XO,YO,ZO,CSEO,SPDO,VL,TWOPI,RDEG,DEGR,TO,TF,FMAX,FMIN,BIN,TT,DSP 0170
     4IF(10,5),IPRI(10,5),IPW(10,5),IPT(10,5),ISCAN(10,5),INEXT(10),JNEXDSP 0180
     5T(10),NUNI,NGAUS,NMTR(10),NOUT(8)                                 DSP 0190
C                                                                       DSP 0200
C                                                                       DSP 0210
C     INITIALIZE CONSTANTS                                              DSP 0220
C                                                                       DSP 0230
C                                                                       DSP 0240
      PI2=ARSIN(1.0)                                                    DSP 0250
      PI=2.*PI2                                                         DSP 0260
      TWOPI=4.*PI2                                                      DSP 0270
      RDEG=180./PI                                                      DSP 0280
      DEGR=PI/180.                                                      DSP 0290
      CON1=PI*58.9                                                      DSP 0300
      CON2=10.*ALOG10(4.*PI)                                            DSP 0310
      TSKTS=0.5626111                                                   DSP 0320
      YDS=2025.4                                                        DSP 0330
      FSCALE=1.E6                                                       DSP 0340
      TSCALE=1.E3                                                       DSP 0350
      PSCALE=1.E-3                                                      DSP 0360
      WSCALE=1.E-6                                                      DSP 0370
      IIJ=0                                                             DSP 0380
      ICT=0                                                             DSP 0390
C                                                                       DSP 0400
C                                                                       DSP 0410
C     SET IFM INFORMATION                                               DSP 0420
C                                                                       DSP 0430
C                                                                       DSP 0440
      READ (5,18) NUNI,NGAUS,VL,TO,TF,TINT,THRESH                       DSP 0450
      VL=VL*FSCALE                                                      DSP 0460
      TINT=TINT*WSCALE                                                  DSP 0470
```

```
      READ (5,19) XO,YO,ZO,CSEO,SPDO,FMAX,FMIN                          DSP 0480
      READ (5,22) NBIN                                                  DSP 0490
      FMAX=FMAX*FSCALE                                                  DSP 0500
      FMIN=FMIN*FSCALE                                                  DSP 0510
      BIN=(FMAX-FMIN)/FLOAT(NBIN)                                       DSP 0520
      WRITE (6,25) XO,YO,ZO,CSEO,SPDO,FMIN,FMAX,BIN,NBIN                DSP 0530
      WRITE (6,26) THRESH                                               DSP 0540
      CSEO=CSEO*DEGR                                                    DSP 0550
      SPDO=SPDO*TSKTS                                                   DSP 0560
      XO=XO*YDS                                                         DSP 0570
      YO=YO*YDS                                                         DSP 0580
      ZO=ZO/3.                                                          DSP 0590
      VXO=SPDO*SIN(CSEO)                                                DSP 0600
      VYO=SPDO*COS(CSEO)                                                DSP 0610
      I=1                                                               DSP 0620
C                                                                       DSP 0630
C                                                                       DSP 0640
C     GET EMITTER PLATFORM INFORMATION                                  DSP 0650
C                                                                       DSP 0660
C                                                                       DSP 0670
    1 READ (5,20) XI,YI,ZI,CSEI,SPDI,NMTR(I)                            DSP 0680
      IF (ZI.LT.0.) GO TO 11                                            DSP 0690
      WRITE (6,27) I,XI,YI,ZI,CSEI,SPDI                                 DSP 0700
      CSEI=CSEI*DEGR                                                    DSP 0710
      SPDI=SPDI*TSKTS                                                   DSP 0720
      VXI=SPDI*SIN(CSEI)-VXO                                            DSP 0730
      VYI=SPDI*COS(CSEI)-VYO                                            DSP 0740
      XI=XI*YDS-XO                                                      DSP 0750
      YI=YI*YDS-YO                                                      DSP 0760
      Z(I)=ZI/3.                                                        DSP 0770
      VTI=SQRT(VXI**2+VYI**2)                                           DSP 0780
      RCSE=ATAN2(VXI,VYI)                                               DSP 0790
      R=SQRT(XI**2+YI**2)                                               DSP 0800
      BNGI=ATAN2(XI,YI)                                                 DSP 0810
      TOF=R/VL                                                          DSP 0820
      X(I)=XI                                                           DSP 0830
      Y(I)=YI                                                           DSP 0840
      VX(I)=VXI                                                         DSP 0850
      VY(I)=VYI                                                         DSP 0860
      BNG(I)=BNGI                                                       DSP 0870
      THETAI=ABS(PI-BNGI+RCSE)                                          DSP 0880
      IF (THETAI.GT.TWOPI) THETAI=THETAI-TWOPI                          DSP 0890
      IF (THETAI.GE.PI2) GO TO 2                                        DSP 0900
      D1=R*COS(THETAI)                                                  DSP 0910
      CPA=R*SIN(THETAI)                                                 DSP 0920
      GO TO 3                                                           DSP 0930
    2 CPA=R                                                             DSP 0940
    3 JJ=NMTR(I)                                                        DSP 0950
```

```
C                                                                       DSP 0960
CCCC                                                                    DSP 0970
CCCC    GET EMITTER INFORMATION                                         DSP 0980
CCC                                                                     DSP 0990
C                                                                       DSP 1000
      DO 10 J=1,JJ                                                      DSP 1010
      READ (5,19) FREQ,PRIN,PULSE,PTRANS,BEAM,GAIN,ANT(I,J)             DSP 1020
      READ (5,21) IFI,IPRII,IPWI,IPTI,ISCANI                            DSP 1030
      PTRANS=PTRANS*TSCALE                                              DSP 1040
      F(I,J)=FREQ*FSCALE                                                DSP 1050
      PRIN=PRIN*PSCALE                                                  DSP 1060
      PRI(I,J)=PRIN                                                     DSP 1070
      PULSE=PULSE*WSCALE                                                DSP 1080
      PW(I,J)=PULSE                                                     DSP 1090
      BW(I,J)=CON1/BEAM                                                 DSP 1100
      POWIJ=10.*ALOG10(PTRANS)+GAIN-CON2+30.                            DSP 1110
      POWER(I,J)=POWIJ                                                  DSP 1120
      WRITE (6,28) J,F(I,J),PRI(I,J),PW(I,J),PTRANS,GAIN,BEAM,POWER(I,J)DSP 1130
      IF (I,J) =IFI+1                                                   DSP 1140
      IPRI(I,J)=IPRII+1                                                 DSP 1150
      IPW(I,J)=IPWI+1                                                   DSP 1160
      ISCAN(I,J)=ISCANI+1                                               DSP 1170
      III=ISCANI+1                                                      DSP 1180
      GO TO (4,5), III                                                  DSP 1190
    4 WRITE (6,29) ANT(I,J)                                             DSP 1200
      GO TO 6                                                           DSP 1210
    5 WRITE (6,30) ANT(I,J)                                             DSP 1220
    6 RMAX=10.**((POWIJ-THRESH)/20.)                                    DSP 1230
C                                                                       DSP 1240
CCCC                                                                    DSP 1250
CCCCC     CALCULATE THE TOA OF THE FIRST PULSE                          DSP 1260
CCC                                                                     DSP 1270
C                                                                       DSP 1280
      IF (RMAX.GE.CPA) GO TO 7                                          DSP 1290
      TNEXT(I,J)=TF+1.                                                  DSP 1300
      WRITE (6,31)                                                      DSP 1310
      GO TO 10                                                          DSP 1320
    7 CALL RANDU (NUNI,IV,RAND)                                         DSP 1330
      NUNI=IV                                                           DSP 1340
      RAND=0.                                                           DSP 1350
      ANG(I,J)=RAND*TWOPI                                               DSP 1360
      CALL RANDU (NUNI,IV,RAND)                                         DSP 1370
      NUNI=IV                                                           DSP 1380
      RAND=0.                                                           DSP 1390
      IF (RMAX.LT.R) GO TO 8                                            DSP 1400
      DEL=0.                                                            DSP 1410
      R1=R                                                              DSP 1420
      GO TO 9                                                           DSP 1430
```

```
      8 D2=CPA*SIN(ARCOS(CPA/RMAX))                                    DSP 1440
        DEL=(D1-D2)/VTI+(R-RMAX)/VL                                    DSP 1450
        R1=RMAX                                                        DSP 1460
      9 TT=TO+PRIN*RAND-TOF-PULSE+DEL                                  DSP 1470
        POW=POWIJ-20.*ALOG10(R1)                                       DSP 1480
        SCNMIN=THRESH-POW                                              DSP 1490
        SIGNAL(I,J)=POW+SCAN(I,J)                                      DSP 1500
        TNEXT(I,J)=TINT*AINT((TT+TOF)/TINT)                            DSP 1510
        SEND(I,J)=TT                                                   DSP 1520
        SIGFRQ(I,J)=F(I,J)                                             DSP 1530
     10 CONTINUE                                                       DSP 1540
C                                                                      DSP 1550
CC                                                                     DSP 1560
CC      FIND THE TOA WITH THE SMALLEST VALUE                           DSP 1570
CC                                                                     DSP 1580
C                                                                      DSP 1590
        I=I+1                                                          DSP 1600
        GO TO 1                                                        DSP 1610
     11 NTGT=I-1                                                       DSP 1620
        WRITE (6,32)                                                   DSP 1630
     12 TEMP=TF                                                        DSP 1640
        L=0                                                            DSP 1650
        ITEMP=0                                                        DSP 1660
C                                                                      DSP 1670
        DO 15 I=1,NTGT                                                 DSP 1680
        JJ=NMTR(I)                                                     DSP 1690
C                                                                      DSP 1700
        DO 14 J=1,JJ                                                   DSP 1710
        TTRY=TNEXT(I,J)                                                DSP 1720
        IF (TEMP.LT.TTRY) GO TO 14                                     DSP 1730
        IF (TEMP.EQ.TTRY) GO TO 13                                     DSP 1740
        TEMP=TTRY                                                      DSP 1750
        ITEMP=I                                                        DSP 1760
        JTEMP=J                                                        DSP 1770
        GO TO 14                                                       DSP 1780
     13 L=L+1                                                          DSP 1790
        INEXT(L)=ITEMP                                                 DSP 1800
        JNEXT(L)=JTEMP                                                 DSP 1810
        ITEMP=I                                                        DSP 1820
        JTEMP=J                                                        DSP 1830
     14 CONTINUE                                                       DSP 1840
C                                                                      DSP 1850
     15 CONTINUE                                                       DSP 1860
C                                                                      DSP 1870
        IF (ITEMP.EQ.0) STOP                                           DSP 1880
C                                                                      DSP 1890
CC                                                                     DSP 1900
C       OUTPUT PULSE DATA                                              DSP 1910
```

```
C                                                                       DSP 1920
C                                                                       DSP 1930
   16 RCVSIG=SIGNAL(ITEMP,JTEMP)                                        DSP 1940
      RCVR=SIGFRQ(ITEMP,JTEMP)                                          DSP 1950
      N=(RCVR-FMIN)/BIN+1.                                              DSP 1960
      RCVBNG=BNG(ITEMP)*RDEG                                            DSP 1970
      IF (RCVBNG.LT.0.0) RCVBNG=RCVBNG+360.                             DSP 1980
      ICT=ICT+1                                                         DSP 1990
      NPW=1.E7*PW(ITEMP,JTEMP)                                          DSP 2000
      TNEXT1=TNEXT(ITEMP,JTEMP)                                         DSP 2010
      NTIME=TNEXT1*1.E7/256.                                            DSP 2020
      NTIME1=MOD(NTIME,65536)                                           DSP 2030
      NTIME2=NTIME/65536                                                DSP 2040
      WRITE (6,23) ICT,N,NTIME2,NTIME1,NPW,TNEXT1                       DSP 2050
      NOUT(IIJ+1)=N                                                     DSP 2060
      NOUT(IIJ+2)=NTIME1                                                DSP 2070
      NOUT(IIJ+3)=NTIME2                                                DSP 2080
      NOUT(IIJ+4)=NPW                                                   DSP 2090
      IIJ=IIJ+4                                                         DSP 2100
      IF (IIJ.LT.8) GO TO 17                                            DSP 2110
      IIJ=0                                                             DSP 2120
      WRITE(7,24) NOUT                                                  DSP 2130
   17 IF (ICT.GE.1024) STOP                                             DSP 2140
C                                                                       DSP 2150
CCCC                                                                    DSP 2160
CCCC  GO CALCULATE THE TOA OF THE NEXT PULSE FROM THIS EMITTER          DSP 2170
CCCC                                                                    DSP 2180
CCC                                                                     DSP 2190
      CALL UPDATE (ITEMP,JTEMP)                                         DSP 2200
      IF (L.EQ.0) GO TO 12                                              DSP 2210
      ITEMP=INEXT(L)                                                    DSP 2220
      JTEMP=JNEXT(L)                                                    DSP 2230
      L=L-1                                                             DSP 2240
      GO TO 16                                                          DSP 2250
C                                                                       DSP 2260
   18 FORMAT (2I10,6E10.0)                                              DSP 2270
   19 FORMAT (8E10.0)                                                   DSP 2280
   20 FORMAT (5E10.0,I1)                                                DSP 2290
   21 FORMAT (10I1)                                                     DSP 2300
   22 FORMAT (I10)                                                      DSP 2310
   23 FORMAT (T40,5I10,1PE20.4)                                         DSP 2320
   24 FORMAT (8(16,1X))                                                 DSP 2330
   25 FORMAT ('1OWN UNIT INFORMATION.',/,'0INITIAL POSITION:',T40,       DSP 2340
     1 'COURSE:',T55,'SPEED:',T70,'FREQUENCY COVERAGE',T100,'BINSIZE',  DSP 2350
     2 T120,'NR BINS',                                                  DSP 2360
     3  /,3F10.1,F16.1,F14.1,1PE20.4,'-',E12.4,E15.4,T120,I7)           DSP 2370
   26 FORMAT (' THRESH HOLD =',F8.1,' DBM')                             DSP 2380
   27 FORMAT ('0TARGET INFORMATION',/'0PLATFORM',T15,'INITIOL POSITION',DSP 2390
```

```
  1 T50,'COURSE',T60,'SPEED',//,I5,T15,3F10.1,T50,F6.0,T60,F5.0,/,    DSP 2400
  2 '0',T10,'EMITTER',T23,'FREQ',T36,'PRI',T49,'PW',T62,'TRANS POWER'DSP 2410
  3,T75,'ANT GAIN',T88,'BEAM WDTH',T101,'EFF POWER',T114,'ANT SCAN')  DSP 2420
                                                                      DSP 2430
28 FORMAT (I14,T20,1P7E13.4)                                          DSP 2440
29 FORMAT ('+',T114,'CIR',F5.2,' SPR')                                DSP 2450
30 FORMAT ('+',T114,'STEADY')                                         DSP 2460
31 FORMAT(' EMITTER OUT OF RANGE')                                    DSP 2470
32 FORMAT('1',T48,'DATA',/,'0',T48,'PULSE',2X,'FREQ BIN',2X,          DSP 2480
  C'DBLE PRECISION TIME',4X,'PW',10X,'TOA')                           DSP 2490
   END
```

```
      SUBROUTINE UPDATE(I,J)                                                DSP 2500
      COMMON X(10),Y(10),Z(10),VX(10),VY(10),BNG(10),F(10,5),TINT,VXO,VYDSP 2510
     1O,THRESH,PI,PI2,SCNMIN,PRIN,SIGFRQ(10,5),PRI(10,5),PW(10,5),POWER(DSP 2520
     210,5),BW(10,5),SEND(10,5),TNEXT(10,5),SIGNAL(10,5),ANT(10,5),ANG(1DSP 2530
     30,5),XO,YO,ZO,CSEO,SPOO,VL,TWOPI,RDEG,DEGR,TO,TF,FMAX,FMIN,BIN,TT,DSP 2540
     4IF(10,5),IPRI(10,5),IPW(10,5),IPT(10,5),ISCAN(10,5),INEXT(10),JNEXDSP 2550
     5T(10),NUNI,NGAUS,NMTR(10)                                             DSP 2560
      PRIN=PRI(I,J)                                                         DSP 2570
      TT=SEND(1,J)+PRIN                                                     DSP 2580
      XI=X(I)                                                               DSP 2590
      YI=Y(I)                                                               DSP 2600
      VXI=VX(I)                                                             DSP 2610
      VYI=VY(I)                                                             DSP 2620
      XX=XI+VXI*TT                                                          DSP 2630
      YY=YI+VYI*TT                                                          DSP 2640
      R=SQRT(XX**2+YY**2)                                                   DSP 2650
      POW=POWER(I,J)-20.*ALOG10(R)                                          DSP 2660
      SCNMIN=THRESH-POW                                                     DSP 2670
      IF (SCNMIN.GE.0.) GO TO 1                                             DSP 2680
      SIGNAL(I,J)=POW+SCAN(I,J)                                             DSP 2690
      TOF=R/VL                                                              DSP 2700
      BNG(I)=ATAN2(XX,YY)                                                   DSP 2710
      TNEXT(I,J)=TINT*AINT((TT+TOF)/TINT)                                   DSP 2720
      SEND(I,J)=TT                                                          DSP 2730
      RETURN                                                                DSP 2740
    1 TNEXT(I,J)=TF+1.                                                      DSP 2750
      WRITE (6,2) I,J                                                       DSP 2760
      RETURN                                                                DSP 2770
C                                                                           DSP 2780
    2 FORMAT (' EMITTER',2I3,'  OUT OF RANGE')                              DSP 2790
      END                                                                   DSP 2800
```

```
      FUNCTION SCAN(I,J)                                                    DSP 2810
      COMMON X(10),Y(10),Z(10),VX(10),VY(10),BNG(10),F(10,5),TINT,VXO,VYDSP 2820
     1O,THRESH,PI,PI2,SCNMIN,PRIN,SIGFRQ(10,5),PRI(10,5),PW(10,5),POWER(DSP 2830
     210,5),BW(10,5),SEND(10,5),TNEXT(10,5),SIGNAL(10,5),ANT(10,5),ANG(1DSP 2840
     30,5),XO,YO,ZO,CSEO,SPDO,VL,TWOPI,RDEG,DEGR,TO,TF,FMAX,FMIN,BIN,TT,DSP 2850
     4IF(10,5),IPRI(10,5),IPW(10,5),IPT(10,5),ISCAN(10,5),INEXT(10),JNEXDSP 2860
     5T(10),NUNI,NGAUS,NMTR(10)                                             DSP 2870
      III=ISCAN(I,J)                                                        DSP 2880
      GO TO (2,1), III                                                      DSP 2890
    1 SCAN=0.0                                                              DSP 2900
      RETURN                                                                DSP 2910
    2 ANGI=ANG(I,J)                                                         DSP 2920
      BWI=BW(I,J)                                                           DSP 2930
      BNGI=BNG(I)                                                           DSP 2940
      RATE=ANT(I,J)                                                         DSP 2950
      RAT1=TWOPI/RATE                                                       DSP 2960
      ANGMIN=1./(BWI*10.**(SCNMIN/20.))                                     DSP 2970
      ANGM=-ANGMIN-PRIN*RAT1                                                DSP 2980
    3 ANGLE=AMOD((ANGI+TT*RAT1-PI-BNGI),TWOPI)                              DSP 2990
      IF (ANGLE.GT.PI) ANGLE=ANGLE-TWOPI                                    DSP 3000
      IF (ANGLE.LT.(-PI)) ANGLE=TWOPI+ANGLE                                 DSP 3010
      IF (ABS(ANGLE).LE.1.E-2) GO TO 1                                      DSP 3020
      IF (ANGLE.GT.ANGMIN) GO TO 4                                          DSP 3030
      IF (ANGLE.LT.ANGM) GO TO 6                                            DSP 3040
      U=BWI*ANGLE                                                           DSP 3050
      SCAN=10.*ALOG10((SIN(U)/U)**2)                                        DSP 3060
      IF (SCAN.GE.SCNMIN) GO TO 7                                           DSP 3070
      TT=TT+PRIN                                                            DSP 3080
      GO TO 3                                                               DSP 3090
    4 DT=(TWOPI+ANGM-ANGLE)/RAT1                                            DSP 3100
    5 TT=TT+(AINT(DT/PRIN)+1.)*PRIN                                         DSP 3110
      GO TO 3                                                               DSP 3120
    6 DT=(ANGM-ANGLE)/RAT1                                                  DSP 3130
      GO TO 5                                                               DSP 3140
    7 RETURN                                                                DSP 3150
      END                                                                   DSP 3160
```

OUTPUT DATA

| PULSE NR | FREQ BIN | DBLE WORD TIME | | PW | TIME |
|---|---|---|---|---|---|
| 1 | 23 | 0 | 0 | 49 | -4.0000E-06 |
| 2 | 39 | 0 | 0 | 23 | -2.0000E-06 |
| 3 | 64 | 0 | 0 | 19 | -1.0000E-06 |
| 4 | 23 | 0 | 48 | 49 | 1.2450E-03 |
| 5 | 23 | 0 | 97 | 49 | 2.4950E-03 |
| 6 | 64 | 0 | 117 | 19 | 2.9980E-03 |
| 7 | 23 | 0 | 146 | 49 | 3.7450E-03 |
| 8 | 23 | 0 | 195 | 49 | 4.9940E-03 |
| 9 | 64 | 0 | 234 | 19 | 5.9980E-03 |
| 10 | 23 | 0 | 243 | 49 | 6.2440E-03 |
| 11 | 23 | 0 | 292 | 49 | 7.4940E-03 |
| 12 | 23 | 0 | 341 | 49 | 8.7440E-03 |
| 13 | 64 | 0 | 351 | 19 | 8.9970E-03 |
| 14 | 23 | 0 | 390 | 49 | 9.9940E-03 |
| 15 | 39 | 0 | 394 | 23 | 1.0088E-02 |
| 16 | 40 | 0 | 429 | 9 | 1.0998E-02 |
| 17 | 23 | 0 | 439 | 49 | 1.1244E-02 |
| 18 | 64 | 0 | 468 | 19 | 1.1997E-02 |
| 19 | 40 | 0 | 472 | 9 | 1.2098E-02 |
| 20 | 29 | 0 | 482 | 40 | 1.2345E-02 |
| 21 | 23 | 0 | 488 | 49 | 1.2494E-02 |
| 22 | 40 | 0 | 515 | 9 | 1.3198E-02 |
| 23 | 29 | 0 | 519 | 40 | 1.3295E-02 |
| 24 | 23 | 0 | 536 | 49 | 1.3744E-02 |
| 25 | 29 | 0 | 556 | 40 | 1.4245E-02 |
| 26 | 40 | 0 | 558 | 9 | 1.4298E-02 |
| 27 | 23 | 0 | 585 | 49 | 1.4994E-02 |
| 28 | 64 | 0 | 585 | 19 | 1.4997E-02 |
| 29 | 39 | 0 | 591 | 23 | 1.5133E-02 |
| 30 | 29 | 0 | 593 | 40 | 1.5195E-02 |
| 31 | 29 | 0 | 630 | 40 | 1.6145E-02 |
| 32 | 23 | 0 | 634 | 49 | 1.6244E-02 |
| 33 | 29 | 0 | 667 | 40 | 1.7095E-02 |
| 34 | 64 | 0 | 703 | 19 | 1.7997E-02 |
| 35 | 29 | 0 | 704 | 40 | 1.8045E-02 |
| 36 | 29 | 0 | 741 | 40 | 1.8995E-02 |
| 37 | 29 | 0 | 779 | 40 | 1.9945E-02 |
| 38 | 39 | 0 | 788 | 23 | 2.0178E-02 |
| 39 | 69 | 0 | 808 | 7 | 2.0697E-02 |
| 40 | 64 | 0 | 820 | 19 | 2.0997E-02 |
| 41 | 23 | 0 | 878 | 49 | 2.2494E-02 |
| 42 | 23 | 0 | 927 | 49 | 2.3744E-02 |
| 43 | 69 | 0 | 928 | 7 | 2.3763E-02 |
| 44 | 64 | 0 | 937 | 19 | 2.3997E-02 |

# BIBLIOGRAPHY

ATAC, Applied Technology Airborne Computer, V. I, II, ITEK Corporation, 1974

Bell, C. G. and Newell, A., Computer Structures: Readings and Examples, McGraw-Hill, 1971

Electronic Countermeasures, U. S. Army Signal Corps, 1961

Gold, B. and Rader, C. M., Digital Processing of Signals, McGraw-Hill, 1969

User's Manual, 1st ed., W. R. Church Computer Center, Naval Postgraduate School, Monterey, California, 1970

INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 3. | Department Chairman, Code 52<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 4. | Professor S. Jauregui, Jr., Code 52Ja<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 10 |
| 5. | Professor G. A. Myers, Code 52Mv<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| 6. | TASES Group, Code 52Ja<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| 7. | Commander Navy Security Group Command (G-80)<br>Navy Security Group Headquarters<br>3801 Nebraska Ave., N.W.<br>Washington, D.C. 20390<br>Attn: CDR Orejuela | 1 |
| 8. | Commander<br>Naval Electronics Systems Command (PME 107)<br>Naval Electronics Systems Command Headquarters<br>Washington, D.C. 20360<br>Attn: LCDR A. Sagarian | 1 |
| 9. | Commander<br>Naval Electronics Systems Command (PME 107)<br>Naval Electronics Systems Command Headquarters<br>Washington, D.C. 20360<br>Attn: LCDR R. Shields | 1 |

10. Director, 1
National Security Agency (W Group)
Fort George G. Mead
Maryland, 20755
Attn: Mr. James Boone

11. Applied Technology 1
645 Almanor Avenue
Sunnyvale, California 94086
Attn: Mr. J. Houston

12. LCDR Paul D. Frazer, USN 1
USS CONE (DD 866)
Fleet Post Office
New York, N.Y. 09501